TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2018 16

Luis Miguel Riazuelo Latas

Mapping and Semantic Perception for Service Robotics

Departamento

Informática e Ingeniería de Sistemas

Director/es

Martinez Montiel, José María
Montano Gella, Luis

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

MAPPING AND SEMANTIC PERCEPTION FOR SERVICE ROBOTICS

Autor

Luis Miguel Riazuelo Latas

Director/es

Martinez Montiel, José María
Montano Gella, Luis

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2018

PhD Thesis

Mapping and Semantic Perception for Service Robotics

Luis Miguel Riazuelo Latas

Advisors

Luis Montano Gella

José María Martínez Montiel

Grupo de Robótica, Percepción y Tiempo Real
Instituto de Investigación en Ingeniería de Aragón
Universidad de Zaragoza

January 2018

To Ainara, Ane and Leo

Acknowledgement

I want to firstly thank my supervisors Luis Montano Gella and José María Martínez Montiel for encouraging me to start this thesis, and for his guidance and support through all these years.

I want also to express my gratitude to Javier Civera whose initial contribution in the development of this thesis was really important. Thanks also for being always available for fruitful discussions. During all these years, I have had the opportunity to participate in several research projects. I would like to thank José Luis Villarroel and Carlos Sagües for the opportunity they gave me to work on the projects under their direction and the trust they have placed in me.

I would also like to thank to all the members of Robotics, Perception and Real Time Group of the University of Zaragoza, and many thanks to all my colleagues in the Robotics lab: Pablo, Oscar, Domenico, Alex, Maite, Marta, Diana, Mayte, Carlos, Eduardo, Yarik, Dorian, Jesús. Special thanks to Anacris and Danilo for adding the constant and necessary pressure during the last weeks of writing this thesis.

Finnaly, I would like to thank to my parents and my brother who have always being there for me. The most special thankfulness is for my wife Ainara who has been the best support in the most stressful moments and for my children Ane and Leo for conveying me their enthusiasm and happiness and for turning every day into a new experience.

Thesis framework

This thesis has been developed with the Robotics, Perception and Real Time group of the Aragón Institute for Engineering Research (I3A), University of Zaragoza, within the framework of the following research projects:

- **Robots móviles en red para tareas de servicio y de intervención (NERO) DPI 2006-07928**
  The complex nature of mobile robot tasks leads to the necessity of systems with several coordinated robots (agents) working in cooperation. Some international directives refer to robotic elements connected to the communication nets or wireless nets including the robots themselves and the sensors distributed in the working place (static agents) exchanging and sharing information. This concept is extended to robot interactions between humans, the sensors and the environment. We propose this project which is very related with previous MEC projects obtained by this research team, to continue working on subjects related to multi-robot cooperation techniques, computer vision, robot vision for motion and communications.

- **Ubiquitous Networking Robotics in Urban Settings (URUS) IST-1-045062-URUS-STP**
  European cities are becoming difficult places to live due to noise, pollution and security. Moreover, the average age of people living European cities is growing and in a short period of time there will be an important community of elderly people. City Halls are becoming conscious of this problem and are studying solutions, for example by reducing the free car circulation areas . Free car areas imply a revolution in the planning of urban settings, for example, by imposing new means for transportation of goods, security issues, etc. In this project we want to analyse and test the idea of incorporating a network of robots (robots, intelligent sensors, devices and communications) in order to improve life quality in such urban areas.

- **TEams of robots for Service and SEcurity missiOns (TESSEO) DPI 2009-08126**
  The project proposes to investigate techniques for a multi-robot team to act in coordination in realistic scenarios. For the deployment, it is necessary to deal with algorithms and methods related to task planning and allocation, coordinated navigation planning, environment perception from multiple views provided by every member of the team, while the communication connectivity among all the elements of the system is maintained – robots, infrastructure, supervisor team, etc. Although some of the techniques involved are usually proposed in the literature and in many projects somehow independently, the research in this project will also be oriented to develop techniques integrating the different subjects involved. Only in this way it will be possible to develop realistic applications using systems with autonomous and supervised behaviours. Within the wide spectrum of scientific and technological challenges that appears in this kind

of systems, several research objectives, grouped in three interdependent blocks, are going to be tackled in this project.

- **Robots sharing a knowledge base for world modelling and learning of actions (RoboEarth) FP7 ICT-248942**
  The aim of RoboEarth is to use the internet to create a giant open source network database that can be accessed and continually updated by robots around the world. With knowledge shared on such a vast scale, and with businesses and academics contributing independently on a common language platform, RoboEarth has the potential to provide a powerful feed forward to any robot's 3D sensing, acting and learning capabilities.

- **Tecnologías Inteligentes para el transporte autónomo de mercancías en interiores y exteriores (TITAM_ie) IDI-20110855**
  The objective is the development of robust technologies for localization, mapping and autonomous navigation of mobile robots for good transportation. A real prototype will be built and the experimental validation will be developed in a large Industrial Park, in indoor and outdoor scenarios.

- **TEams of robots for LOgistics, MAintenance and eNvironment monitoring (TELOMAN) DPI 2012-32100**
  The research project involves deployment and actuation techniques of a multi-robot team. It is necessary to address problems of task planning and allocation, coordinated execution of the navigation, perception of the environment from multiple views from each of the team members, maintaining communication between all system components – robots, infrastructure, bridges, monitoring equipment, etc. This project will address new goals and challenges for research and their application in real, large and complex scenarios.

- **Debris removal in tunnels through robotized dumpers (AUTODUMP) RTC-2015-4099-4**
  The objective of this project is to design and develop a new kit to robotize a conventional dumper used in construction, transforming it to an autonomous mobile robot for tunnel construction. It must be capable of reaching the excavation front without human intervention, then wait to be loaded, and finally autonomously transport the debris outside of the tunnel towards the dump. The new robotic kit will be a breakthrough in the technology used to carry out work performances of various kinds. In addition, the kit developed also represents a major technological challenge to maximize the autonomy of the process and its ability to react in a dynamic and minimally structured environment such as a tunnel in construction. These factors besides offering a marketable product, require the development of different subsystems involved to ensure robustness, compactness and economic viability of the system.

- **Robot exploration and navigation in challenging environments (ROBOCHALLENGE) DPI2016-76676-R-AEI/FEDER-UE**
  This project addresses theoretical and experimental research in the field of intervention and exploration of challenging environments by means of aerial and ground robots. A challenging environment is one where, due to different reasons, current robotic techniques fail or do not work properly and continuously. This failure is due to the characteristics of the environment, such as lack of coverage GNSS (any environment confined), absence of visual or geometric discriminant characteristics (tunnels, pipes), environments with sections of different characteristics (tunnel construction, mine), complex patterns of communication signal propagation (side

galleries, caves), dynamic environments (presence of workers or other vehicles), absence of quality in lighting (tunnel construction, mine, pipeline, cave), or large (any real environment).

- **Robotic Testbed in an ARt and Technology center (RT-ART) H2020- 645220-RAWFIE**
  The overall goal of RT-ART is to provide a realistic environment for ground robot experimentation. This will be achieved by our contributed means integrated within the RAWFIE project sharing infrastructure. Five different scenarios will be available within the ETOPIA: the large museum entrance, an exhibition hall, a large gallery and connected corridors, a residential area and an outdoor terrace, in which four UGV will be available. Monitoring tools, prior maps and assistance with the experimentation will be provided according to RAWFIE infrastructure.

Resumen

Para realizar una tarea, los robots deben ser capaces de ubicarse en el entorno. Si un robot no sabe dónde se encuentra, es imposible que sea capaz de desplazarse para alcanzar el objetivo de su tarea. La localización y construcción de mapas simultánea, llamado SLAM, es un problema estudiado en la literatura que ofrece una solución a este problema. El objetivo de esta tesis es desarrollar técnicas que permitan a un robot comprender el entorno mediante la incorporación de información semántica. Esta información también proporcionará una mejora en la localización y navegación de las plataformas robóticas. Además, también demostramos cómo un robot con capacidades limitadas puede construir de forma fiable y eficiente los mapas semánticos necesarios para realizar sus tareas cotidianas.

El sistema de construcción de mapas presentado tiene las siguientes características: En el lado de la construcción de mapas proponemos la externalización de cálculos costosos a un servidor en nube. Además, proponemos métodos para registrar información semántica relevante con respecto a los mapas geométricos estimados. En cuanto a la reutilización de los mapas construidos, proponemos un método que combina la construcción de mapas con la navegación de un robot para explorar mejor un entorno y disponer de un mapa semántico con los objetos relevantes para una misión determinada.

En primer lugar, desarrollamos un algoritmo semántico de SLAM visual que se fusiona los puntos estimados en el mapa, carentes de sentido, con objetos conocidos. Utilizamos un sistema monocular de SLAM basado en un EKF (Filtro Extendido de Kalman) centrado principalmente en la construcción de mapas geométricos compuestos únicamente por puntos o bordes; pero sin ningún significado o contenido semántico asociado. El mapa no anotado se construye utilizando sólo la información extraída de una secuencia de imágenes monoculares. La parte semántica o anotada del mapa -los objetos- se estiman utilizando la información de la secuencia de imágenes y los modelos de objetos precalculados.

Como segundo paso, mejoramos el método de SLAM presentado anteriormente mediante el diseño y la implementación de un método distribuido. La optimización de mapas y el almacenamiento se realiza como un servicio en la nube, mientras que el cliente con poca necesidad de computo, se ejecuta en un equipo local ubicado en el robot y realiza el cálculo de la trayectoria de la cámara. Los ordenadores con los que está equipado el robot se liberan de la mayor parte de los cálculos y el único requisito adicional es una conexión a Internet.

El siguiente paso es explotar la información semántica que somos capaces de generar para ver cómo mejorar la navegación de un robot. La contribución en esta tesis se centra en la detección 3D y en el diseño e implementación de un sistema de construcción de mapas semántico.

A continuación, diseñamos e implementamos un sistema de SLAM visual capaz de funcionar con robustez en entornos poblados debido a que los robots de servicio trabajan en espacios compartidos con personas. El sistema presentado es capaz de enmascarar las zonas de imagen ocupadas

por las personas, lo que aumenta la robustez, la reubicación, la precisión y la reutilización del mapa geométrico. Además, calcula la trayectoria completa de cada persona detectada con respecto al mapa global de la escena, independientemente de la ubicación de la cámara cuando la persona fue detectada.

Por último, centramos nuestra investigación en aplicaciones de rescate y seguridad. Desplegamos un equipo de robots en entornos que plantean múltiples retos que implican la planificación de tareas, la planificación del movimiento, la localización y construcción de mapas, la navegación segura, la coordinación y las comunicaciones entre todos los robots. La arquitectura propuesta integra todas las funcionalidades mencionadas, asi como varios aspectos de investigación novedosos para lograr una exploración real, como son: localización basada en características semánticas-topológicas, planificación de despliegue en términos de las características semánticas aprendidas y reconocidas, y construcción de mapas.

Abstract

In order to perform a task, robots need to be able to locate themselves in the environment. If a robot does not know where it is, it is impossible for it to move, reach its goal and complete the task. Simultaneous Localization and Mapping, known as SLAM, is a problem extensively studied in the literature for enabling robots to locate themselves in unknown environments. The goal of this thesis is to develop and describe techniques to allow a service robot to understand the environment by incorporating semantic information. This information will also provide an improvement in the localization and navigation of robotic platforms. In addition, we also demonstrate how a simple robot can reliably and efficiently build the semantic maps needed to perform its quotidian tasks.

The mapping system as built has the following features. On the map building side we propose the externalization of expensive computations to a cloud server. Additionally, we propose methods to register relevant semantic information with respect to the estimated geometrical maps. Regarding the reuse of the maps built, we propose a method that combines map building with robot navigation to better explore a room in order to obtain a semantic map with the relevant objects for a given mission.

Firstly, we develop a semantic Visual SLAM algorithm that merges traditional with known objects in the estimated map. We use a monocular EKF (Extended Kalman Filter) SLAM system that has mainly been focused on producing geometric maps composed simply of points or edges but without any associated meaning or semantic content. The non-annotated map is built using only the information extracted from an image sequence. The semantic or annotated parts of the map –the objects– are estimated using the information in the image sequence and the precomputed object models.

As a second step we improve the EKF SLAM presented previously by designing and implementing a visual SLAM system based on a distributed framework. The expensive map optimization and storage is allocated as a service in the Cloud, while a light camera tracking client runs on a local computer. The robot's onboard computers are freed from most of the computation, the only extra requirement being an internet connection.

The next step is to exploit the semantic information that we are able to generate to see how to improve the navigation of a robot. The contribution of this thesis is focused on 3D sensing which we use to design and implement a semantic mapping system.

We then design and implement a visual SLAM system able to perform robustly in populated environments due to service robots work in environments where people are present. The system is able to mask the image regions occupied by people out of the rigid SLAM pipeline, which boosts the robustness, the relocation, the accuracy and the reusability of the geometrical map. In addition, it estimates the full trajectory of each detected person with respect to the scene global map, irrespective of the location of the moving camera at the point when the people were imaged.

Finally, we focus our research on rescue and security applications. The deployment of a multi-

robot team in confined environments poses multiple challenges that involve task planning, motion planning, localization and mapping, safe navigation, coordination and communications among all the robots. The architecture integrates, jointly with all the above-mentioned functionalities, several novel features to achieve real exploration: localization based on semantic-topological features, deployment planning in terms of the semantic features learned and recognized, and map building.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics has experienced almost unbelievable progress in the last 50 years. Robots now form part of our working lives in factories and industries. In the near future, robots will probably enter our homes and form part of our daily lives, as predicted by many science fiction movies. They will be used in different areas (see figure 1.1) including domestic tasks, personal transportation, security and surveillance, field robotics, rescue operations, etc.

In order to perform a task, robots need to be able to locate themselves in the environment. If a robot does not know where it is, it is impossible for it to be able to move around and achieve the goal of its task. SLAM, standing for Simultaneous Localization and Mapping, is an extensively-studied problem in the literature. The goal of SLAM algorithms is to solve the problems of placing a mobile robot in an unknown environment and position and ensure that it is able to build a consistent map of the environment incrementally while using that map to determine its own location. Although they produce an accurate localization, the maps are mainly composed of features without meaning (points, lines). Adding semantic information such as ”objects” or ”people” with referenced locations on the map built by the robot allows a better understanding of the environment than the metric location provided by featured SLAM methods. The richer and more complex the environment model is, the more useful it becomes for a robot in order to perform autonomous tasks.

The ability to efficiently create semantic environment models and to use them intelligently to locate objects will become increasingly important as more and more robots enter human living and working environments. To successfully operate in such environments, robots will have to face the *open-world challenge*, i.e. they will need to be able to handle large numbers of (novel) objects located in various places, for example on top of or inside furniture, and they will need to quickly become acquainted with novel environments. This poses several challenges for today’s robots, for example: How can a robot efficiently explore an environment to create a map of the objects therein? What are the most important objects to look out for? How can the robot exploit the semantic information to guide a search? How can the visual perception system handle large numbers of object models without slowing down recognition or detecting more false positives? How can it profit from information collected by other robots? We believe that finding solutions to these problems will be crucial to scale object search tasks from restricted and well-known laboratory environments to more open and diverse scenes.

The aim of this thesis is to enable a service robot to understand the environment by incorporating semantic information. This information will also provide an improvement in the localization and navigation of robotic platforms. In addition to a better understanding of the environment, it will

Figure 1.1.: Service robots in different scenario applications.

provide the possibility of interacting with it.

1.1. Service Robotics: Concept and Architecture

Service robotics is the area addressing the development of robots that are capable of interacting with the environment and with the humans within it. The function of a service robot is to assist a human being in the performance of different tasks that can be dangerous, repetitive, dirty, or dull for a person. These robots can be autonomous or teleoperated remotely by means of a central control station.

Service robots for personal/domestic use	Service robots for professional use	
Robots for domestic tasks	Field robotics	Rescue & security applications
Entertainment robots	Professional cleaning	Defense applications
Elderly and handicap assistance	Inspection and maintenance systems	Underwater systems
Personal transportation (AGV for persons)	Construction and demolition	Powered Human Exoskeletons
Home security & surveillance	Logistic systems	Unmanned aerial vehicles
Other Personal / domestic robots	Medical robotics	

Table 1.1.: Classification of a service robot by application areas.

The International Federation of Robotics (IFR) has proposed a definition for a service robot: ”A service robot is a robot which operates semi- or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations”. Service robots are designed to interact with people and different areas. The classification of a service robot for

personal/domestic use or for professional use is done according to its intended application. Table 1.1 summarizes the application areas. Furthermore, in order to be able to perform its tasks in a given environment, the robot must have several capabilities (figure 1.2):

- **Mapping:** the ability to generate a model of the environment and establish a frame of reference for its later use.
- **Localization:** to establish its own position and orientation within the frame of reference.
- **Perception:** the capability of sensing the environment and acquiring and processing information from it.
- **Path planning:** the ability to search and get control strategies to obtain from the robot suitable and safe trajectories of the highest quality in its movement.
- **Obstacle avoidance:** to ensure obstacle-free navigation. The robot must be able to react in real time to the obstacles that appear in its dynamic environment.
- **Communications:** enables data sharing among robots, coordination with other robots, and even human use.

Figure 1.2.: Components that implements the main capabilities of a service robots.

1.2. Contributions and organization of the thesis

This thesis has been developed within the framework of different research projects. This has given us the ability to understand what components are necessary for a robotic platform to move in an environment and how they have to be integrated. We have also been able to see the strengths and weaknesses of each of the basic components integrated into the software architecture of a service robot. In this thesis we have focused our contributions on the introduction of semantic information in SLAM maps, and the exploitation of such information for robot navigation. The core of the thesis is the development of visual SLAM systems and the incorporation of semantic information into the generated maps. On the map building side, we propose mapping methods able to build accurate geometrical maps efficiently in mobile devices by the externalization of expensive computations to a cloud server. Additionally, we propose methods to register relevant semantic information such as people, objects or places with respect to the estimated geometrical maps. Maps not only have to be built, they also have to be used by autonomous robots to control their interaction with the environment. We propose a method that combines map building with robot navigation to better explore a room in order to obtain a semantic map with the relevant objects for a given mission. The evaluation of this proposal has been carried out in office and hospital room environments. Furthermore, thanks to the research projects in which we have been working throughout the development of this thesis, we have been able to apply the same semantic and distributed mapping principles to a team of robots for intervention in confined environments. The main blocks and contributions of this thesis are detailed below, as well as a brief introduction to them. The related publications are also detailed in each block, adding references to section 1.2.1 where a summary of all the contributions can be found.

Map composed of sparse points and objects

The first step of this thesis presented in chapter 2 is to develop a semantic Visual SLAM algorithm that merges in the estimated map traditional meaningless points with known objects (see figure 1.3). We use a monocular EKF-SLAM system that has been mainly focused on producing geometric maps composed simply of points or edges, but without any associated meaning or semantic content. The

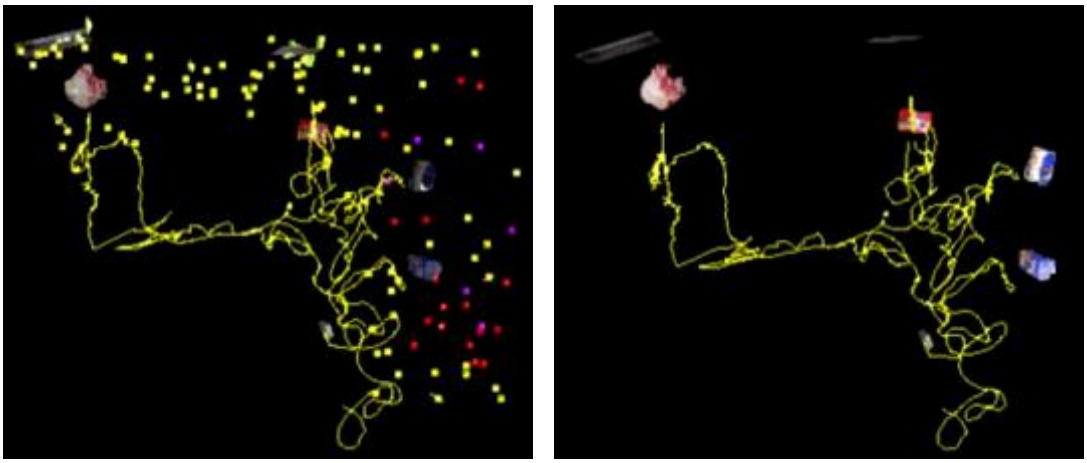


Figure 1.3.: Left: Features map an camera trajectory computed by a classical visual SLAM algorithm. Right: known objects detected in the scene and introduced in the map.

known object models are automatically computed from a sparse set of images gathered by cameras that may be different from the SLAM camera, creating a database of objects. The contribution in this work is to design and implement a system that runs an EKF monocular SLAM parallel to an object recognition thread. The semantic or annotated parts of the map –the objects– include the estimated position for each object within the map using the information in the image sequence and the precomputed object models. This object detector algorithm informs about the presence of an object in the sequence by searching for SURF correspondences and checking afterwards their geometric compatibility. When an object is recognized it is inserted in the SLAM map, its position being measured and refined by the SLAM algorithm in subsequent frames. This was one of the first mapping systems able to provide object locations in real time. However, the mapping system, based on EKF, scales poorly with the map size, limiting its size to a few hundred features. Similarly, the search for the objects in the database was based on an exhaustive search for matches between the current image and all the images defining the objects in the database, which also scaled poorly with the database size. For this reason, in chapters 3 and 4 we can see how these scaling limitations are overcome. The contribution of this work was reported in [C1] and [W4]. In [V1] and [V2] we can see videos of the experimental validation.

Figure 1.4.: Two camera clients building a cooperative map of the same environments using C^2TAM algorithm.

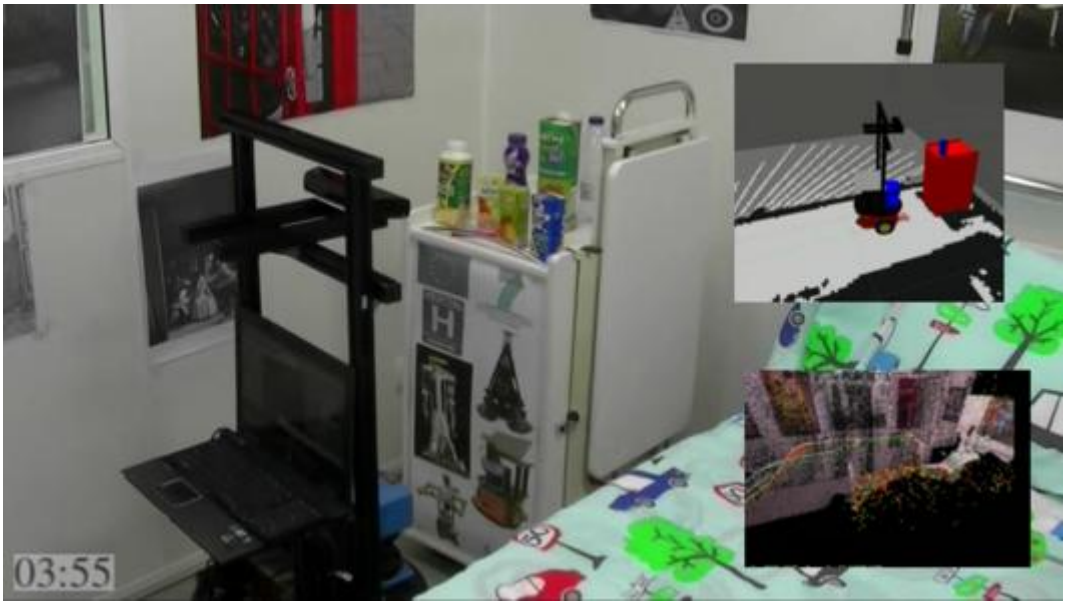


Figure 1.5.: Action recipe execution by a service robot. Right upper image presents the semantic map and Octomap generated, and features and 3D reconstruction is showed in right lower image.

Cooperative mapping in the cloud

Running a SLAM algorithm by an autonomous mobile robot is a computationally demanding process for medium and large-scale scenarios, in spite of the progress made both in the algorithmic and hardware areas. As a consequence, a robot with SLAM capabilities has to be equipped with the latest computers whose weight and power consumption might limit its autonomy. In chapter 3 we present the second contribution in the development of this thesis. We have designed and implemented a visual SLAM system based on a distributed framework where the expensive map optimization and storage is allocated as a service in the Cloud, while a light camera tracking client runs on a local computer. Additionally, the system is able to build a map cooperatively using two or more robots. Figure 1.4 shows a cooperative mapping execution where two camera clients T1 and T2 are building a common map M of an office. Unlike the method used in the previous work presented in chapter 2, this algorithm is based on keyframes and provides a significant improvement in terms of the accuracy and size of the estimated map. The robot's onboard computers are freed from most of the computation, the only extra requirement being an internet connection. The experimental validation is focused on showing real-time performance for a single-robot and cooperative SLAM using an RGBD camera. The system provides the interface to a map database where: 1) a map can be built and stored, 2) stored maps can be reused by other robots, 3) a robot can fuse its map online with a map already in the database, and 4) several robots can estimate individual maps and fuse them together if an overlap is detected. Regarding the map size, we can handle maps up to 7000 points while maintaining real time performance. This contribution was reported in [J1], [W1] and [W3]. In [V3] and [V4] we can see videos of the experimental validation. We also release the software related to this contribution in [S1].

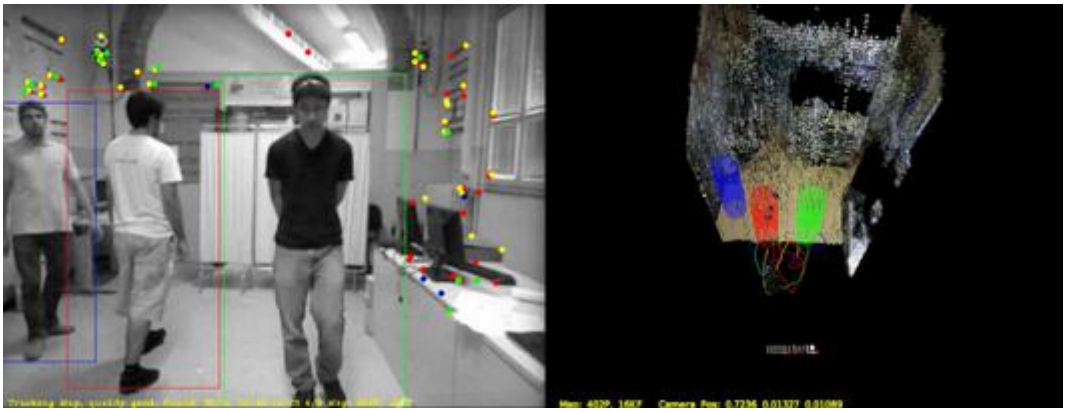


Figure 1.6.: Left: Features extracted on the image and people detections. Right: 3D map reconstruction using depth information and people instances and its trajectories.

Semantic Mapping System: A Cloud Enabled Knowledge-Based Approach

Once we have seen the importance of semantic maps in chapter 3, we can exploit the semantic information to improve the navigation of a robot. In chapter 4, we summarize the cloud mapping facilities developed within the RoboEarth project *, which aims to design a knowledge-based system to provide web and cloud services that can transform a robot into an intelligent one. Our contribution is focused on 3D sensing, designing and implementing a semantic mapping system. The system improves the proof of concept presented in chapter 2 in two fundamental aspects. On the one hand, the visual SLAM algorithm presented in the previous section (detailed in chapter 3) is used in contrast to the monocular EKF with the corresponding improvement in accuracy and map size. Secondly, the object detection algorithm has been improved and in this system a Bag of Words object recogniser is used with the corresponding boost in the database size from a few tens to a few hundreds or even thousands of objects. The semantic map is composed of (1) an ontology to code the concepts and relations in maps and objects, and (2) a SLAM map providing the scene geometry and the object locations with respect to the robot. We propose to ground the terminological knowledge in the robot perceptions by means of the SLAM map of objects. We demonstrate how a simple robot can reliably and efficiently build the semantic maps needed to perform its quotidian tasks. In addition, we show the synergetic relation of the SLAM map of objects that grounds the terminological knowledge coded in the ontology. For validating the system, we investigate two *action recipes* that embody semantic map building in a simple mobile robot (see figure 1.5). The first recipe enables semantic map building for a novel environment (a hospital room) while exploiting available prior information about the most likely objects present in those environments, looking for relevant and easy to find objects in the room which, once inserted in the map, can help in finding other objects more difficult to find due to occlusion. The second recipe searches for a new object, with the efficiency boosted by reducing the time spent on searching for the object by means of a directed search, thanks to the reasoning in a previous semantic map of the room. This contribution was reported in [J2] and [W2]. In [V5] we can see a video of the experimental validation.

*http://roboearth.ethz.ch

Semantic Visual SLAM in Populated Environments

In contrast to the case presented in previous work (chapter 4), most service robots work in environments where the people are presented. In chapter 5, we propose a visual SLAM system able to perform robustly in populated environments. The image stream from a moving RGB-D camera is the only input to our system. The main contributions of this work are: 1) improving the SLAM system performance when it is working in populated scenarios; 2) providing semantic information about the people's activity, which can be used by the robot to plan or execute its tasks. A real-time human tracker is embedded into the system. Figure 1.6 shows the RGB image with features extracted by the SLAM algorithm, the bounding box of people detected and 3D map reconstruction with people instances and their trajectories in the map frame. The computed map in real-time is composed of two layers: 1) The unpopulated geometrical layer, which describes the geometry of the bare scene as an occupancy grid where pieces of information corresponding to people have been removed; 2) A semantic human activity layer, which describes the trajectory of every person with respect to the map, labelling areas as "traversable" or "occupied". The system has been evaluated using the visual SLAM algorithm presented in chapter 3 and another state of the art visual SLAM algorithm based on keyframes. The pipeline of the approach presented is as follows: first, to mask the image regions occupied by people out of the rigid SLAM pipeline, which boosts the robustness, the relocation, the accuracy and the reusability of the geometrical map in populated scenes. Secondly, to estimate the full trajectory of each detected person with respect to the scene map, irrespective of the location of the moving camera when the person was imaged. People are recognized in the scene and are inserted into the generated SLAM map, in the same way as we inserted the objects in the works presented in chapter 2 and chapter 4. This contribution was reported in [C2]. In [V6] we can see a video of the experimental validation.

Service robotics in confined and structured environments

One of the applications of service robotics, as shown in table 1.1, focuses on intervention, rescue, or security applications. As mentioned above, the participation in several research projects whose experimentation was performed in these kinds of environments gave us the opportunity to apply the same concepts of mapping and semantic perception presented in chapter 4 to safety, security and rescue robotics. The works described in chapters 2-5 are oriented mainly to the semantic interpretation of scenes. Chapter 6 is devoted to developing a complete robotic system in which semantic scene interpretation is integrated to other robotic functionalities. Furthermore, in chapters 2-5 the semantic information handled were people and objects, while in chapter 6 we focus on the search for places. Deploying a multi-robot team in such confined environments poses multiple challenges that involve task planning, motion planning, localization and mapping, safe navigation, coordination and communications among all the robots. Our contribution is focused on three aspects: localization based on semantic-topological features, deployment planning in terms of the semantic features learned and recognized, and map building. In contrast to previous works, in this approach we use localization algorithms based on lidar sensor data. However, the same principle based on the cloud robotics paradigm presented in chapter 3 is used where the map building is carried out on the server side, in this case at the central station. We also produce a high-level plan of instructions similar to the action recipes introduced in chapter 4, where navigation based on semantic information is presented. Figure 1.7 shows an example of a high-level plan based on relevant places (crosses, intersections, corridors) in the scenario, which are detected and recognized, and the map built by the robot team. This contribution was reported in [J3] and all the related contributions presented in the field of autonomous

navigation and localization in [J4, J5, B1, B2, B3, B4]. In [V6] and [V7] we can see videos of the experimental validation. Lessons learned from many experiments carried out in these scenarios are reported as a useful contribution in the robotics field.

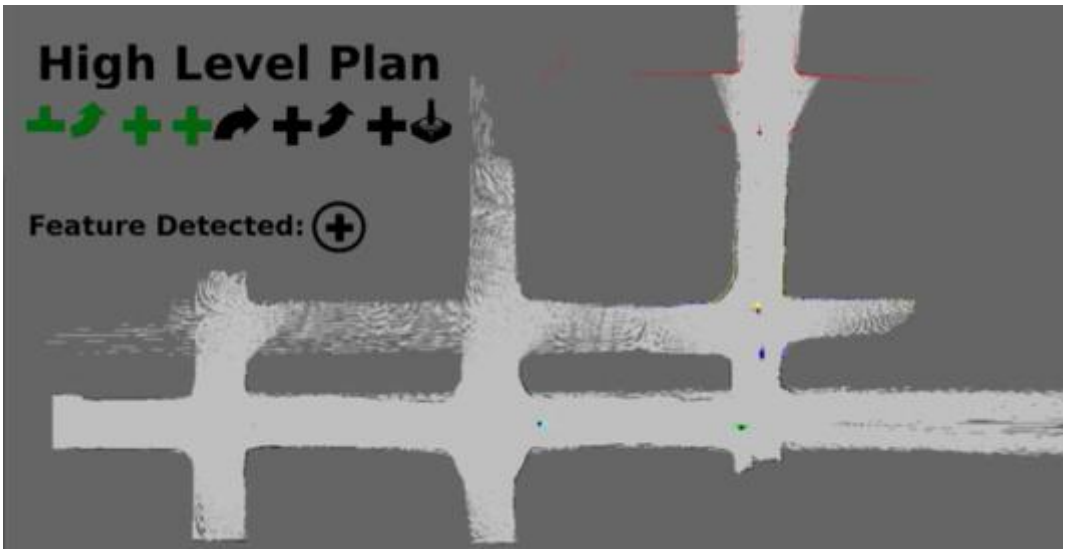


Figure 1.7.: Robot deployment in a maze-like environment, corresponding to a real mine (Santa Marta, Toledo) and high level plan performed.

1.2.1. Contributions

This work is composed of the following publications

Journal articles

- **[J1]** Riazuelo et al. (2014b). **C^{2}TAM: A Cloud Framework for Cooperative Tracking and Mapping**. L. Riazuelo, Javier Civera, J. M. M. Montiel. Robotics and Autonomous Systems (RAS). April 2014, vol. 62(4), pp. 401-413.

- **[J2]** Riazuelo et al. (2015). **RoboEarth Semantic Mapping: A Cloud Enabled Knowledge-Based Approach**. Luis Riazuelo, Moritz Tenorth, Daniel Di Marco, Marta Salas, Dorian Gálvez-López, Lorenz Mösenlechner, Lars Kunze, Michael Beetz, Juan D. Tardós, Luis Montano, J. M. M. Montiel. IEEE Transactions on Automation Science and Engineering (T-ASE). Special Issue on Cloud Robotics and Automation. April 2015, vol. 12(2), pp. 432-443.

- **[J3]** Tardioli et al. (2016). **Robot teams for intervention in confined and structured environments**. Danilo Tardioli, Domenico Sicignano, Luis Riazuelo, Antonio Romeo, José Luis Villarroel, Luis Montano. Journal of Field Robotics (JFR), 2016. Special Issue on Safety, Security, and Rescue Robotics (SSRR). March 2016.

International Conferences

- **[C1]** Civera et al. (2011). **Towards Semantic SLAM using a Monocular Camera**. J. Civera, D. Gálvez-López, L. Riazuelo, J. D. Tardos, J.M.M Montiel. 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2011), San Francisco, California, 2011.

- **[C2]** Riazuelo et al. (2017). **Semantic Visual SLAM in Populated Environments**. Luis Riazuelo, Luis Montano and J. M. M. Montiel. The European Conference on Mobile Robotics (ECMR 2017), Paris (France), 2017.

International and National Workshops

- **[W1]** Riazuelo et al. (2014a). **Real-Time 3D Reconstruction using the Cloud**. L. Riazuelo, Javier Civera and J. M. M. Montiel. III Jornada de Jóvenes Investigadores del I3A. Instituto Universitario de Investigación en Ingeniería de Aragón, Universidad de Zaragoza - Spain, 29 May 2014.

- **[W2]** Riazuelo et al. (2013). **RoboEarth Web-Enabled and Knowledge-Based Active Perception**. L. Riazuelo, M. Tenorth, D. Di Marco, M. Salas, L. Mösenlechner, L. Kunze, M. Beetz, J. D. Tardos, L. Montano, J. M. M. Montiel. Workshop on AI-based Robotics. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), Tokyo - Japan, 3-8 November 2013.

- **[W3]** Riazuelo et al. (2013). **C^{2}TAM: A First Approach to a Cloud framework for Cooperative Tracking And Mapping**. L. Riazuelo, Javier Civera and J. M. M. Montiel. Workshop on Cloud Robotics: Online Knowledge Bases, Web Services, and Cloud Computing for Robots. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), Tokyo - Japan, 3-8 November 2013.

- **[W4]** Marco et al. (2012). **Creating and using RoboEarth object models**. Daniel Di Marco, Andreas Koch, Oliver Zweigle, Kai Häussermann, Björn Schießle, Paul Levi, Dorian Gálvez-López, Luis Riazuelo, Javier Civera, J. M. M. Montiel, Moritz Tenorth, Alexander Perzylo, Markus Waibel and René van de Molengraft. 2012 IEEE International Conference on Robotics and Automation (ICRA2012, Video Proceedings), Saint Paul (Minnesota, USA), 2012.

1.2.2. Related contributions

Throughout the completion of this thesis, we have participated in various projects and collaborated on research topics related to autonomous navigation and localization. The results obtained can be seen in the following publications

- **[J4]** Tardioli et al. (2010). **Enforcing network connectivity in robot team missions**. Danilo Tardioli, Alejandro R. Mosteo, Luis Riazuelo, José L. Villarroel and Luis Montano. International Journal of Robotics Research (IJRR). January 2010, vol. 29(4), pp.460-480.
- **[J5]** Rizzo et al. (2013). **Signal Based Deployment Planning for Robot Teams in Tunnel-like Fading Environments**. Carlos Rizzo, Danilo Tardioli, Domenico Sicignano, Luis Riazuelo, José L. Villarroel and Luis Montano. International Journal of Robotics Research (IJRR). September 2013, vol. 32(12), pp. 1381-1397.
- **[B1]** Rizzo et al. (2015). **Guaranteeing communication for robotic intervention in long tunnel scenarios**. Carlos Rizzo, Domenico Sicignano, Luis Riazuelo, Danilo Tardioli, Francisco Lera, José Luis Villarroel and Luis Montano. In Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, Victor Muñoz-Martinez (Editors). ROBOT 2015: Second Iberian Robotics Conference. Advances in Intelligent Systems and Computing. Volume 1, 2015.
- **[B2]** Tardioli et al. (2017). **A robotized dumper for debris removal in tunnels under construction**. Danilo Tardioli, Luis Riazuelo, Teresa Seco, Jesús Espelosín, Jorge Lalana, José Luis Villarroel, Luis Montano. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau, Carlos Cardeira (Editors). ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing. Volume 1, 2017.
- **[B3]** Barrios et al. (2017). **Low-Bandwidth Telerobotics in Fading Scenarios**. Samuel Barrios, Natalia Ayuso, Danilo Tardioli, Luis Riazuelo, Alejandro R. Mosteo, Francisco Lera and José Luis Villarroel. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau, Carlos Cardeira (Editors). ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing. Volume 2, 2017.
- **[B4]** Abad et al. (2017). **Integrating an Autonomous Robot on a Dance and New Technologies Festival**. Paula Abad, Miguel Franco, Rosa Castillón, Iñigo Alonso, Ana Cambra, Jorge Sierra, Luis Riazuelo, Luis Montano and Ana Cristina Murillo. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau, Carlos Cardeira (Editors). ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing. Volume 1, 2017.

1.2.3. Open-Source Software

We have released the following open-source software:

- **[S1]** C^2TAM (`https://github.com/lriazuelo/c2tam`)
  A Cloud framework for cooperative tracking and mapping

1.2.4. Videos

Demonstrating videos of map composed of sparse points and objects:

- **[V1]** Desktop sequence: `https://youtu.be/kAiGgQwI684`
- **[V2]** Hospital room sequence: `https://youtu.be/emk1b9WTXAk`

Demonstrating videos of C^2TAM:

- **[V3]** Desktop sequence: `https://youtu.be/kE5wmFoCV5E`
- **[V4]** Office sequence: `https://youtu.be/giMDnKhkg-0`

Demonstrating videos of RoboEarth semantic mapping system:

- **[V5]** Semantic mapping action recipes: `https://youtu.be/ehVt-eqk3dI`

Demonstrating videos of semantic visual SLAM in populated environments:

- **[V6]** SLAM masking people: `https://youtu.be/HKe1-kXtM_E`

Demonstrating videos of service robotics in confined and structured environments:

- **[V7]** Santa Marta mine: `https://youtu.be/ZjLNDHzji4Q`
- **[V8]** Somport tunnel: `https://youtu.be/BpxHXTZjKF4`

Chapter 2

Towards Semantic SLAM using a Monocular Camera

The first step of this thesis presented in chapter 2 is to develop a semantic Visual SLAM algorithm that merges in the estimated map traditional meaningless points with known objects (see figure 1.3). We use monocular EKF-SLAM system that have been mainly focused on producing geometric maps just composed of points or edges; but without any associated meaning or semantic content. The known object models are automatically computed from a sparse set of images gathered by cameras that may be different from the SLAM camera. The contribution in this work is to design and implement a system that runs an EKF monocular SLAM parallel to an object recognition thread. The semantic or annotated part of the map –the objects– are estimated using the information in the image sequence and the precomputed object models. This object detector algorithm informs of the presence of an object in the sequence by searching for SURF correspondences and checking afterwards their geometric compatibility. When an object is recognized it is inserted in the SLAM map, being its position measured and hence refined by the SLAM algorithm in subsequent frames. As proof of concept we realized of the potential of introducing semantic information into maps. However, the SLAM algorithm used as well as the object recognition algorithm could be improved. For this reason, in chapters 3 and 4 we can see how this improvement has been carried out. The contribution of this work was reported in [C1] and [W4].

2.1. Introduction

The goal of the work presented in this chapter is to enrich usual monocular SLAM maps by partially annotating the geometric estimation with object information. The map produced by most of the SLAM algorithms in the literature was a joint estimation of geometric entities –in most cases points or lines– without any semantic meaning or annotations attached to it.

The addition of semantic content to a geometric SLAM map is done by recognizing object instances and registering them into the estimated map. We have chosen monocular vision as the only input for constructing the object models, recognizing them and estimating the geometric map. We believe there are tree reasons why such a minimalist configuration has a high potential for robotic applications: 1) there is a large body of recognition models from the computer vision community; 2) the Internet is widely populated with pictures that could serve for automatically generating the recognition models in the future; 3) different cameras can be used at different steps. Cameras for model

building can be different for cameras used in SLAM –so they are in our experiments–, what provides interoperability to exploit the same models by diverse robots. In A we present the way to build up and use an extensive sensor-independent object model database.

The contribution presented in this work is a proof of concept of semantic Visual SLAM using a monocular vision sensor. The integrated system proposed combines three different algorithms: First, Monocular EKF- provides robust real-time camera location and a sparse map of salient point features Civera et al. (2010). Second, Structure from Motion is used to precompute a database of object models from sparse images Snavely et al. (2008). Third, Visual Recognition allows to detect the presence of the object in an image stream Martinez et al. (2010); Hinterstoisser et al. (2010).

There are three main issues that lead us to the combination of local mapping plus object recognition and registration as a feasible solution in a robotic mapping problem. Firstly, a robot has to interact with its environment, hence the 3D registration of the object within the environment is needed to perform any task. The sole presence of the object in an image is not enough in most of the robotic applications (e.g., grasping an object). Secondly, the usual visual input of most robots is not a single image but a sequence, and hence the recognition algorithms should exploit this rich sensorial input. Finally, every robotic task poses severe real-time constraints. Using again the grasping example, we are interested in the location of the robotic arm with respect to the object *at the specific grasp moment*. The objects are registered within the SLAM map as soon as they are recognized, then their 3D location is available at any moment afterwards. For example, once an object is recognized and inserted in the map, the robot will be able to locate it for grasping even when it is out of its field of view.

Summing up, the main motivation for our work is the vision that the combination of object recognition and local SLAM using images –what we call Semantic Monocular SLAM– forms a basic sensing capability suitable for task execution by diverse robots. Object recognition algorithms on their own are unable to register several objects in a common reference frame. Any standard geometric SLAM algorithm can provide an accurate geometric registration, but in its own cannot be used for high-level task definition (for example, 'grasp a cup'). This problem will be addressed in depth in chapter 4.

The rest of the chapter is organised as follows: Section 2.2 discusses related work and section 2.3 summarizes the whole algorithm. Next sections are devoted to further details about the different components of the algorithm: section 2.4 to the object model, section 2.5 to the object recognition and 2.6 to the backbone point-based monocular SLAM where objects are registered into. Finally, section 2.7 shows experimental results and section 2.8 concludes and presents lines for future work.

2.2. Related work

Visual recognition has been used in robotic SLAM mostly for *scene recognition*, for example loop closing in Cummins and Newman (2008) or Cadena et al. (2010) and relocalization in Williams et al. (2007). But it has been very scarcely combined with SLAM for *object recognition*. Image-based recognition was used in Ekvall et al. (2006); Meger et al. (2008); Zender et al. (2008) to register known objects into a laser SLAM map. Differently from these latest three works, we do not only use the appearance from a single image but the appearance and the estimated geometry from a set of images of the object.

The closest approach to this work is the work by Castle et al. Castle et al. (2007, 2010), which registers planar objects into an EKF-based monocular SLAM. They use SIFT features to construct the appearance model of the objects and insert in the SLAM map three of the boundary corners of the plane. As our main contribution over this work, we are able to overcome the planar restriction and

can deal with any object geometry. Our object model is composed by the appearance; but also by the geometric position of salient point features over the object. When the object is recognized, we insert in the SLAM map the recognized SURF points using general –non-planar– geometry constraints.

As the word *semantic* may have a broad meaning, it is worth remarking than in this approach it is referred to the labeling of certain map features in a SLAM map. Although we find it a promising line of research, we do not consider here yet the relations between objects in a scene (as done, for example, in Ranganathan and Dellaert (2007)).

2.3. Notation and general overview

Given that geometric SLAM is typically faster than object recognition, our algorithm is divided into two threads: one dedicated to monocular SLAM and other one to object recognition. Figure 2.1 gives an general overview of the algorithm using figures from our experimental results. Our proposal is briefly summarized here and further detailed in the remaining sections.

Let start at step $k-m$. The image I_{k-m} is used both in the monocular SLAM and the recogntion threads. The monocular SLAM thread uses this image to update the geometric state $\mathbf{x}$ from the previous step $k-m-1$ to the current one $k-m$ using a standard EKF formulation. At the same time, the SLAM state vector is augmented with the current camera pose. Such augmentation will be necessary every time the recognition thread starts for a coherent object insertion: the recognition results will arrive at step k; but still the object insertion should be made with respect to image I_{k-m}. The details about the state augmentation are described in section 2.6.3.

We have a model for every object we want to recognize $O_1,\ldots,O_p,\ldots,O_q$ stored in a database. Each one of these models contains appearance and geometric information extracted from a set of images of the object. The appearance information is composed of SURF descriptors extracted from the images; and the geometric information is the 3D position of the SURF points.

The object recognition thread inserts in the SLAM map the objects in the database as follows: First, the algorithm searches for correspondences between the image I_{k-m} and the images of the objects $O_1,\ldots,O_p,\ldots,O_q$. SURF point features are extracted from image I_{k-m} and compared with the precomputed SURF descriptors for each object. If the number of correspondences exceed a threshold, the geometric compatibility of the matches is checked using RANSAC. If most of the candidates are shown to be geometrically compatible, they are finally assumed to belong to the object and inserted into the SLAM map. Once in the SLAM map, these points are tracked and hence its position refined in the rest of the sequence.

We will assume in this work that there are not repeated objects nor false object detections (in fact, false object detections did not appear in any experiment with the system). Nevertheless, our opinion is that the proposed combination could easily handle both situations. As objects in our system are geometrically registered, any repeated object detection in a different location than the previous one is a different object of the same class. The geometric object registration also serves for the case of false detections: as the SLAM algorithm keeps measuring the object features once inserted, any geometric inconsistency produced by a false detection would be rejected as spurious by the 1-point RANSAC.

2.4. Object model

Our model for an object O is based on the information extracted from a set of images I_l of it. In order to construct the object model, SURF features Bay et al. (2008) are extracted in first place from the set of images I_l. The SURF descriptors $\mathbf{d}^l_O$ form the appearance model for the object. The geometric position

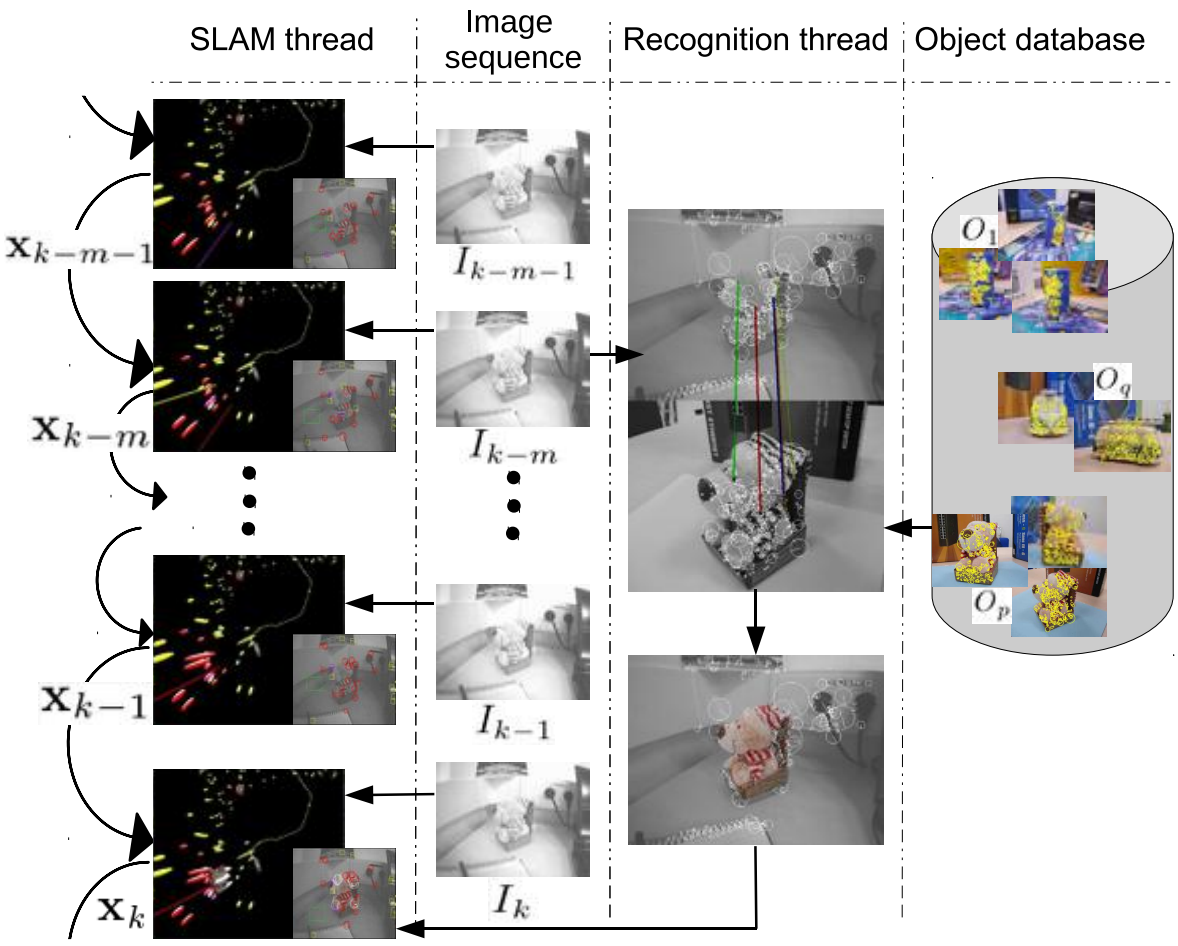


Figure 2.1.: Overview of the algorithm using figures from our experiments.

for the SURF points $\mathbf{y}_O$ is computed using pairwise Structure from Motion algorithms Hartley and Zisserman (2004): correspondences are searched over the images I_l using the SURF descriptors, the geometry is estimated between pairs of images using robust algorithms, and a non-linear optimization step finally produces the 3D estimation for SURF points $\mathbf{y}_O$ and the relative motion between the images I_l. A modern software library called *Bundler* Snavely et al. (2008) has been used in our work for constructing the model.

As standard cameras usually do not capture a whole object in an image –you never see the front and the back of an object at the same time–, the object model is divided into *faces*. We name *face F* to the tuple $< \mathbf{t}_F^O, \mathbf{q}_F{}^O, \mathbf{y}_F^O, \mathbf{d}_F >$. That means each *face* corresponds to an image augmented with the appearance –the SURF descriptors $\mathbf{d}_F$– and the geometric information –the 3D position of the SURF points $\mathbf{y}_F^O$– needed for the recognition. The *face* also includes the position $\mathbf{t}_F^O$ and orientation $\mathbf{q}_F^O$ of the image with respect to an object reference frame O. The object reference frame has been chosen to be the centroid of the 3D point cloud with two axis aligned with its principal components.

Apart from the model for each face F, a global dense point cloud 3D model is also constructed starting from the *faces* and the estimated location of the object images. Multi-View Stereo algorithms (specifically the software package *PMVS* Furukawa and Ponce (2010)) are used for this purpose. It

should be remarked that this dense model is not used for object 3D registration nor recognition –which are done by the appearance and position of the SURF features. We think a dense model could be of importance for robotic applications like grasping; but in this work this dense model is only used for visualization purposes.

Figure 2.2 shows the object model for a *teddy bear*, used in our first experiment. Three representative images out of the twenty that were used to construct the model are displayed. Each one of these images is the basis for a *face*. The SURF features that will be used for recognition are drawn in the images as white circles. Finally, it is also displayed the dense 3D model that will be inserted in the SLAM map.

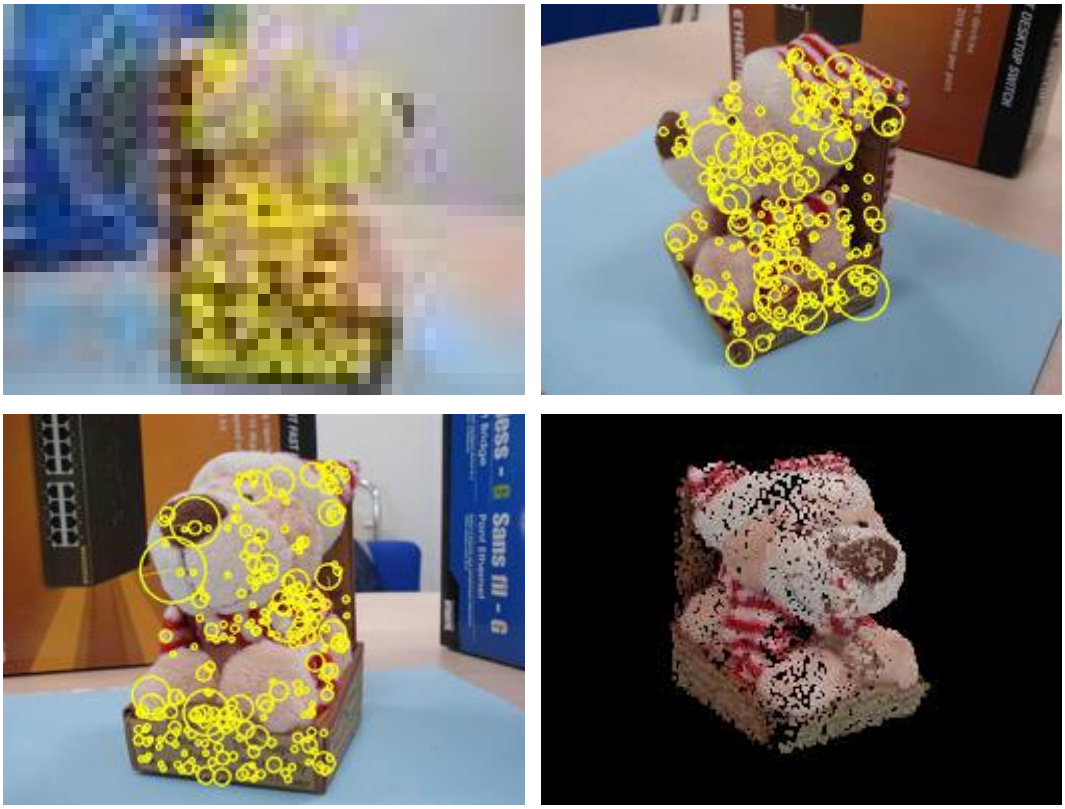


Figure 2.2.: The top row and bottom left images show 3 out of the 20 images that were used for the model of the teddy bear in our first experiment. Notice the SURF features used for recognition superimposed over the images. The bottom right image shows the dense 3D model that will be registered in the SLAM map.

2.5. Object recognition

The object recognition algorithm starts by extracting SURF features from the image I_{k-m}. For each object O in the database, correspondences are calculated between I_{k-m} and its *faces* F by applying the distance ratio (NNDR) to fast approximate nearest neighbors Muja and Lowe (2009). These correspondences are then checked to be geometrically consistent. The RANSAC algorithm is run to find a subset of at least 5 correspondences, between different SURF features, that describe a valid transformation between the SLAM image I_{k-m} and the object image I_F. The Perspective-n-Point

(PnP) problem Moreno-Noguer et al. (2007) is used to estimate the translation $\mathbf{t}_F^{C_{k-m}}$ and orientation $\mathbf{q}_F^{C_{k-m}}$ that define this transformation from the image features and 3D points. However, for planar objects, an homography is estimated instead, with the DLT algorithm Hartley and Zisserman (2004).

The correspondences which are not consistent with the transformation or the homography are rejected. If we obtain a valid transformation between I_{k-m} and some F, we stop searching the rest of the faces of the object O and consider this found.

When a face $F \equiv < \mathbf{t}_F^O,\ \mathbf{q}_F{}^O,\ \mathbf{y}_F^O,\ \mathbf{d}_F >$ of an object O is detected, the final pose of the object is obtained by refining the estimated transformation with only those correspondences which were inliers. The 3D points $\mathbf{y}_F^O$ corresponding to the matches $\mathbf{d}_F$ are then fed into the monocular SLAM system for object insertion; and are inserted as detailed in section .

2.6. Monocular SLAM

In this section the additions to the standard mode EKF are described.

2.6.1. Standard Mode EKF

We follow the 1 point RANSAC EKF proposed in Civera et al. (2010). The estimated parameters are modeled as a multidimensional Gaussian variable $\mathbf{x}$, including camera motion parameters at step k $\mathbf{x}_{C_k}$ and the n point features $\mathbf{y}_i$ that form the map. The geometry of the detected objects will be added to this standard system.

$$\mathbf{x}_k = \left(\begin{array}{ccccccc} \mathbf{x}_{C_k}^\top & \mathbf{y}_1^\top & \cdots & \mathbf{y}_i^\top & \cdots & \mathbf{y}_n^\top \end{array} \right)^\top . \tag{2.1}$$

Camera parameters $\mathbf{x}_{C_k}$ at step k include camera position $\mathbf{t}_{C_k}$ and orientation $\mathbf{q}_{C_k}$ and also linear and angular velocities $\mathbf{v}_{C_k}$ and ω_{C_k}. Point features $\mathbf{y}_i$ are initialized using inverse depth parametrization and are converted to Euclidean coordinates $\mathbf{y}_i^\top = (X\ Y\ Z)$ when projection equation shows a high degree of linearity.

Robust data association relies on 1-point RANSAC EKF, proven to be an efficient implementation of a spurious rejection into an EKF framework; presenting low cost even for large number of matches and large rates of spurious data.

2.6.2. State Augmentation with Past Camera Pose

When the recognition thread is started, the monocular SLAM state has to be augmented with camera location at this step. Using the notation from figure 2.1, at step $k-m$ the EKF step is augmented by copying camera position $\mathbf{t}_{C_{k-m}}$ and orientation $\mathbf{q}_{C_{k-m}}$ into the state vector and propagating covariances accordingly. At step $k-1$, just before the object insertion, the state vector will then be as follows:

$$\mathbf{x}_k = \left(\begin{array}{ccccccc} \mathbf{x}_{C_{k-1}}^\top & \mathbf{y}_1^\top & \cdots & \mathbf{y}_n^\top & \mathbf{t}_{C_{k-m}}{}^\top & \mathbf{q}_{C_{k-m}}{}^\top \end{array} \right)^\top . \tag{2.2}$$

Such augmentation will last until the recognition thread finishes, which usually will take several EKF steps. If an object from the database has been recognized the past camera location will be used for the delayed initialization of the recognized object. In any case, after the recognition has finished the augmentation will not be needed and the past camera position and orientation $\mathbf{t}_{C_{k-m}}$ and $\mathbf{q}_{C_{k-m}}$ will be marginalized out from the state.

2.6.3. Object Insertion

The output of the recognition thread is a set of points y_F corresponding to SURF points of the object in the *face* F, which is the image where the recognition was started –it is worth remembering that do not correspond to the current image I_k. This 3D position y_F of the SURF features is available as a part of the object model described in section 2.4. Also, the translation $\mathbf{t}_F^{C_{k-m}}$ and rotation $\mathbf{q}_F^{C_{k-m}}$ between the face F and the SLAM image I_{k-m} has been computed by the PnP algorithm.

The object points $\mathbf{y}_F^O$ are transformed to the SLAM reference W according to the following formula:

$$\mathbf{y}_F^W = \mathbf{H}_{C_{k-m}}^W \left(\mathbf{t}_{C_{k-m}}, \mathbf{q}_{C_{k-m}}\right) \mathbf{H}_O^{C_{k-m}} \mathbf{y}_F^O \ . \tag{2.3}$$

$\mathbf{t}_{C_{k-m}}$ and $\mathbf{q}_{C_{k-m}}$ are the position and orientation of the SLAM camera when the object recognition was initiated and was stored in the state vector $\mathbf{x}$ as described in section . $\mathbf{H}_O^{C_{k-m}}$ represents the motion between the object reference frame O and the SLAM camera C_{k-m} at step $k-1$. It can be computed by composition:

$$\mathbf{H}_O^{C_{k-m}} = \mathbf{H}_F^{C_{k-m}} \left(\mathbf{t}_F^{C_{k-m}}, \mathbf{t}_F^{C_{k-m}}\right) \mathbf{H}_O^F \left(\mathbf{t}_O^F, \mathbf{q}_O^F\right) \ . \tag{2.4}$$

Point covariances are propagated accordingly, and points $\mathbf{y}_F^W$ in the W reference are finally inserted into the SLAM map.

$$\mathbf{x}_k \ = \ \left(\ \mathbf{x}_{C_k}^\top \quad \mathbf{y}_1^\top \quad \dots \quad \mathbf{y}_n^\top \quad \mathbf{y}_F^{W\top} \ \right)^\top \ . \tag{2.5}$$

After its insertion, the object points will be measured using the standard pinhole camera model over Euclidean points. The reader is referred to Civera et al. (2008b) for further details on the measurement equation.

2.6.4. Relocalization

In the practical use of any monocular SLAM system, tracked features may be lost due to varied reasons: a dynamic object covering most of the image, lost images or sudden motions producing blur. A relocation system able to recover the camera location with respect to a previous map is essential for any practical system. In our approach, we are using the relocation system in Williams et al. (2007).

2.7. Experimental results

The two experiments presented here were recorded with the same camera, a low-cost black-and-white Unibrain camera with a resolution of 320×240. The model of the objects were built from images taken with a standard consumer digital camera. The image sequences for both experiments were gathered at 30 frames per second and used at this frequency as the input to our experiments. As the computational cost of the proposed algorithm is higher than 33 milliseconds for the map sizes used, some of the frames may be skipped by the Semantic SLAM.

2.7.1. Desktop Environment

The sequence for this experiment has 8951 frames and was recorded by moving hand held camera over a desktop in one of our laboratories for about 5 minutes. The lighting conditions are particularly

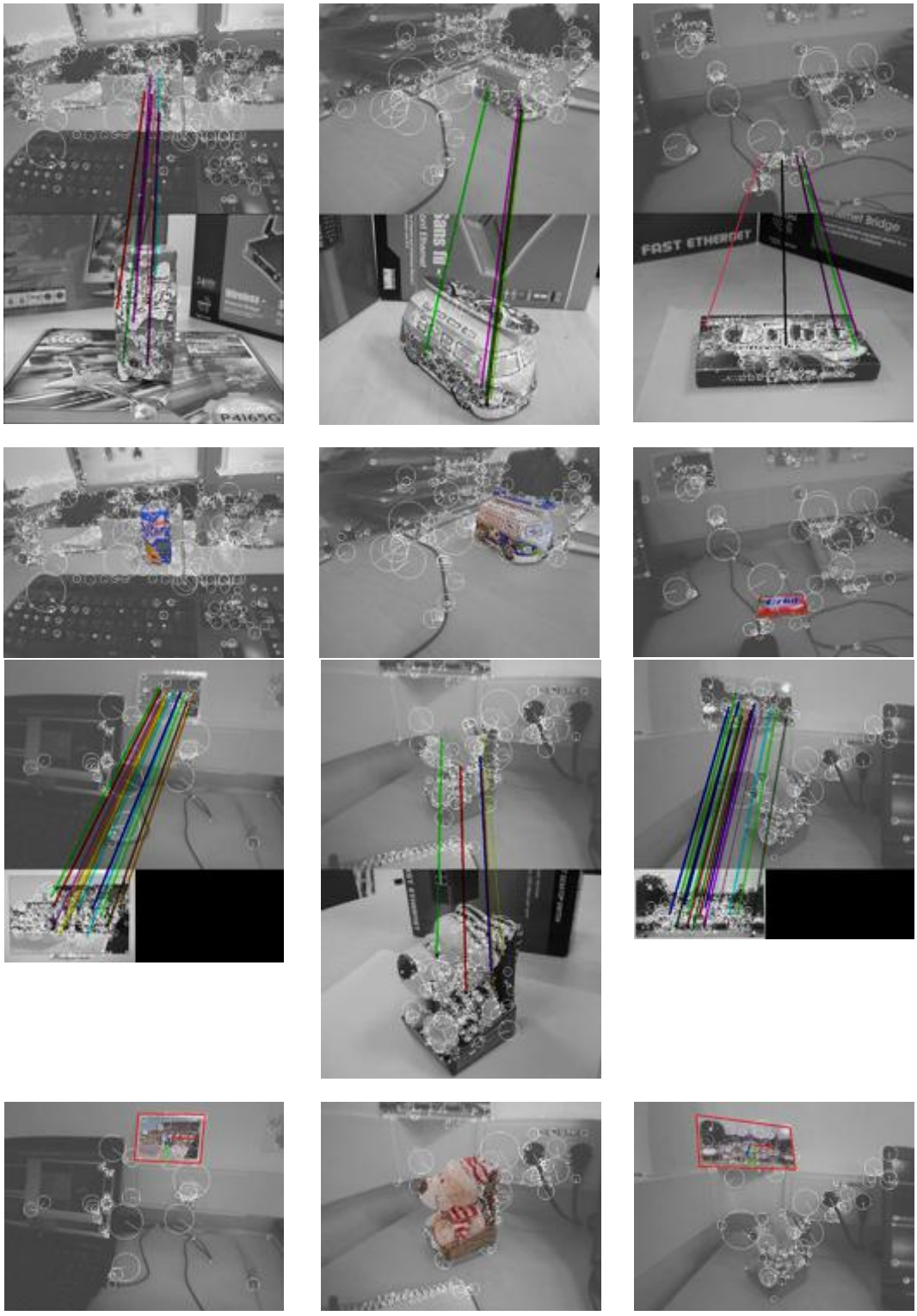


Figure 2.3.: Object recognition thread results, showing columnwise the specific frames where the six objects were detected. The top row shows the image in the monocular sequence input to the Semantic SLAM, the middle row shows the *face* of the object model, and the coloured lines are the matches. The object is inserted in the map based on those correspondences. The bottom row shows the dense point cloud of the object over the image, proving the correct alignment.

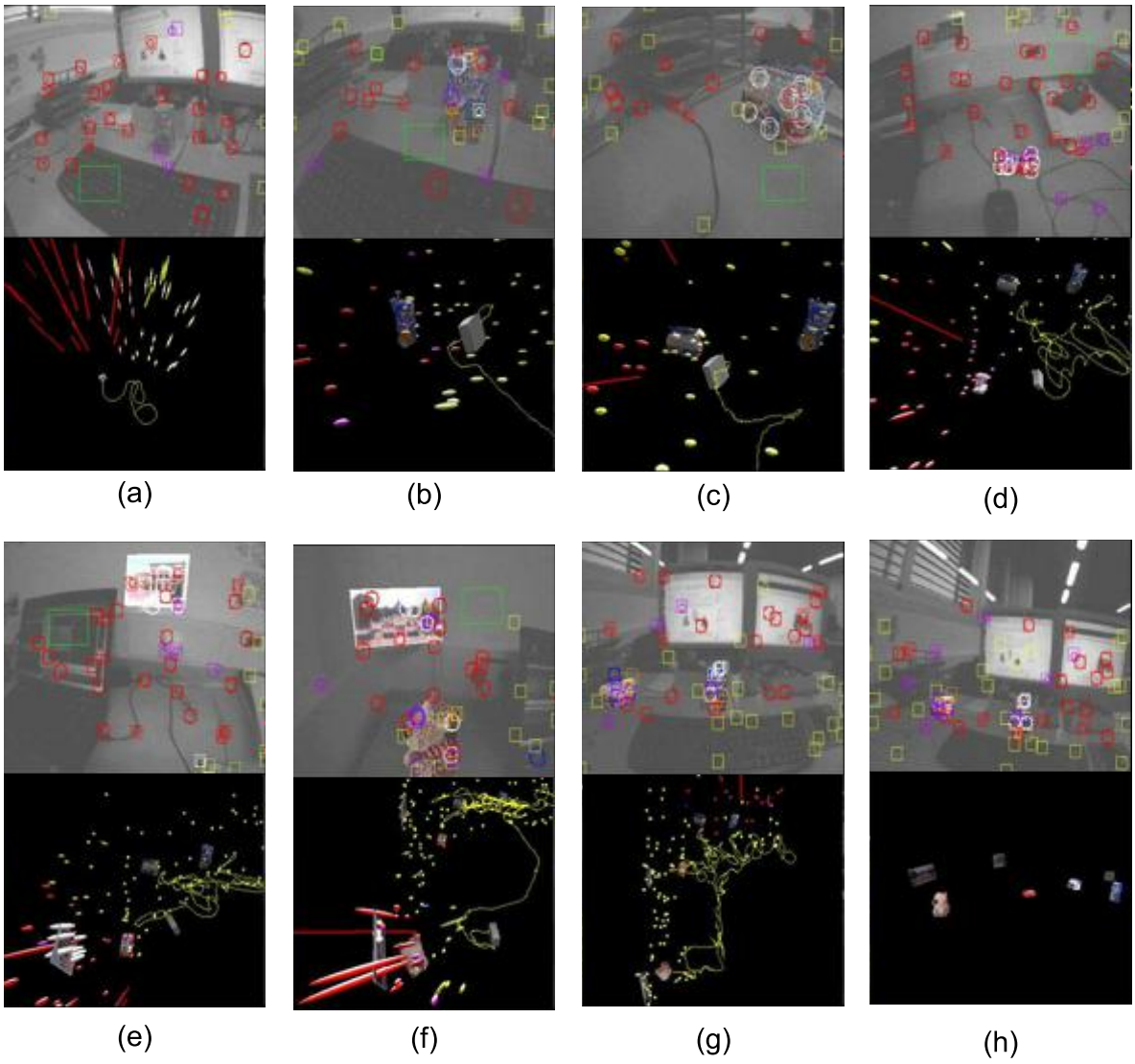


Figure 2.4.: Representative images (at the top) and 3D estimation at their respective times (at the bottom) for the desktop experiment. (a) Initial map, still with no recognized objects. (b) The tetra pack has been recognized, inserted and is being tracked. (c), (d) and (e): the Volkswagen replica, chewing gum packet and the postcard has been inserted. (g) The two remaining objects –postcard and teddy bear– are inserted. (h) The camera moves back close to the starting position, revisiting all previous objects. (h) Final frame of the sequence and 3D view of the objects registered in a common reference frame.

bad for this sequence, as it can be observed in figure 2.4. The desktop contains four 3D objects –a *tetra pack* of fruit juice, a replica of a *Volkswagen replica*, a packet of *chewing gum* and a *teddy bear*– and two planar objects –two *postcards*. The planar models of the two postcards where built from a single image each of them, and the models of the other four objects were constructed from several images taken around each of them, following the technique described in section 2.4.

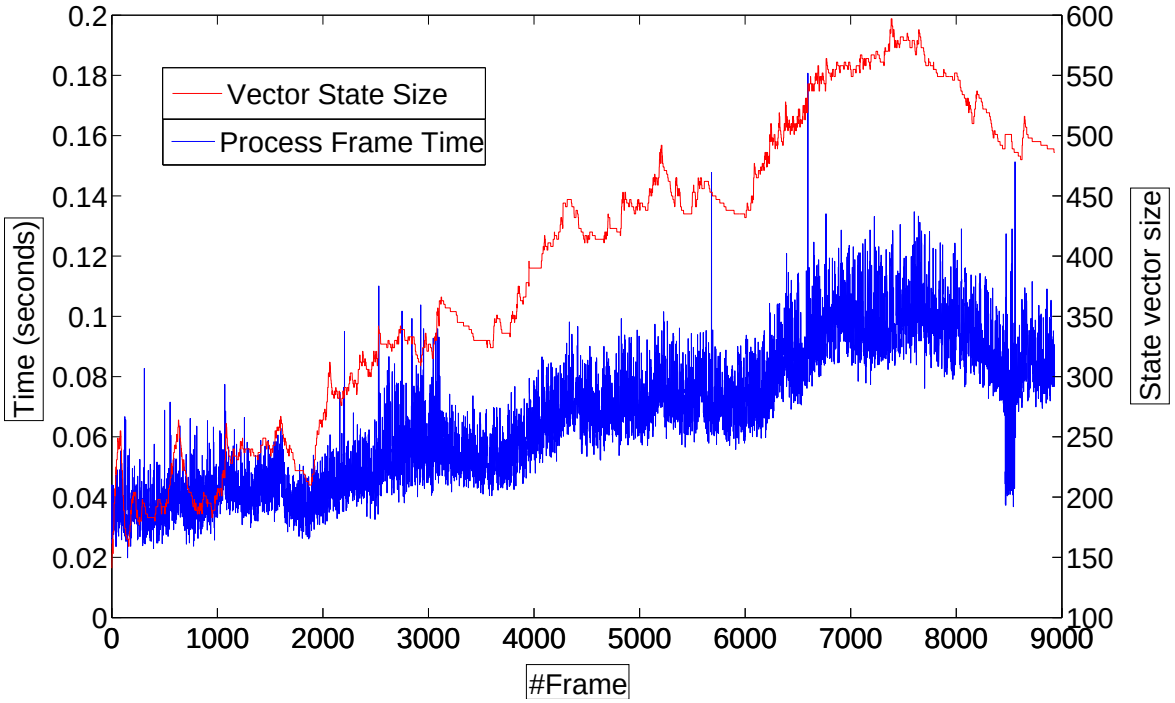


Figure 2.5.: Computational cost (in blue) and state vector size (in red) for the experiment. Notice that the algorithm runs, in the worst case, at around 7 Hz for a state vector of size 600.

All the six objects that compose our database are detected along the sequence, inserted in the 3D map and tracked the rest of the sequence. Figure 2.4 summarizes the results of this experiment for several steps of the estimation. For each step it is shown the current frame and a 3D view of the estimation. The 3D view show the uncertainty ellipses for each point in the 3D map, the dense point clouds modeling the objects and the camera trajectory as a yellow line. The tracked points are also displayed over the current frame –an ellipse shows the search area and a square frames the actual match– in different colours: red stands for succesfully tracked points, blue for those rejected by low patch cross-correlation, magenta for those rejected by our 1-point RANSAC, white for points succesfully tracked in known objects and orange for points not matched in a known object.

Figure 2.3 shows the results for the recognition thread at the specific frames where the six objects were recognized. The top row shows the frame in the sequence where they were recognized; the middle row the object *face* that was recognized; and the coloured lines stand for the correspondences between the two. In the bottom row it is displayed the reprojection of the dense point cloud model.

Figure 2.4.a shows the estimation results at step #610, when no object has been inserted yet. Figure 2.4.b shows frame #1359, just after the tetra pack has been detected and inserted. Figures 2.4.c, 2.4.d and 2.4.e show respectively that the *Volkswagen replica*, the *chewing gum* packet and the *postcard* have been inserted in the map and are being tracked. Their correspondent frames in the

Figure 2.6.: Representative images (at the top) and 3D estimation at their respective times (at the bottom) for the hospital room experiment. The tracked features are displayed in the images as circles. The inserted objects, represented as coloured prismatic solids, are also reprojected at the images. The 3D views show the camera trajectory up to this frame as a yellow line, the point feature uncertainties as ellipses and the inserted objects. (a), frame #34, the cabinet has been already recognized and inserted. (b), frame #241, the robot goes forward. Notice that, although the cabinet is not seen in the image, its 3D position remains registered. (c), frame #912, the robot turns left and faces the cabinet and the tetra pack, which is detected and registered. (d), the robot turns, recognizing and registering the bed. (e) Robot location at the end of the experiment.

sequence are #2764, #4062, #4725. Figure 2.4.f shows the results at step #7102, when the two latest objects inserted –the *teddy bear* and the *postcard*– have been inserted. In figure 2.4.g the camera has gone back to the starting point (frame #8538), imaging again the *tetra pack* and the *Volkswagen replica*. Finally, figure 2.4 is the last frame of the sequence, showing only the objects in the SLAM map. We strongly recommend to watch the video * for a better understanding of this experiment.

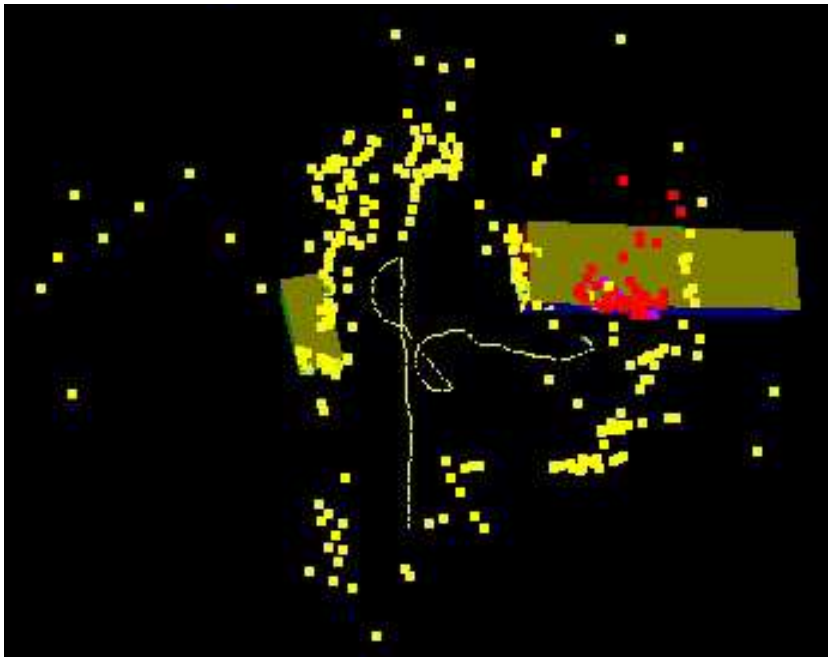
Finally, figure 2.5 shows the computational time (blue line) along with the state vector size (red line) for this experiment. The experiment was run in an Intel Core i7 at 2.66 GHz. In the worst case, the proposed algorithm runs at 7 Hz for a state size of around 600.

2.7.2. Hospital Room Environment –RoboEarth Project

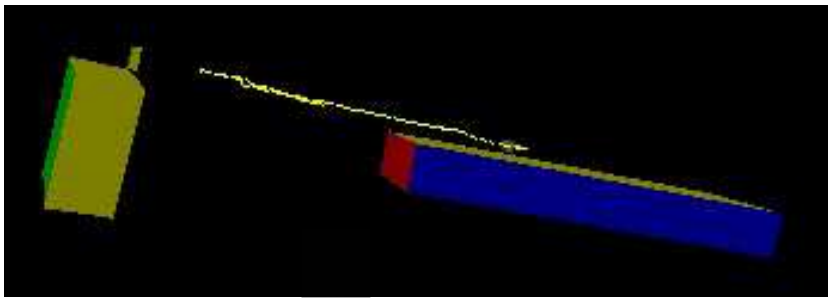
Before focusing on the visual SLAM part, it is worth explaining the general aim of this experiment. The Unibrain camera is mounted on an holonomous robot that moves inside a hospital room environment. This experiment is framed into the RoboEarth project Waibel et al. (2010); dedicated to construct an Internet-like database for robots to share knowledge. In chapter 4 we will show more details about this project. When the robot enters the room, it downloads from the RoboEarth database the recognition object models and an *action recipe* that commands it to serve the tetra pack to the patient in bed. The Semantic SLAM algorithm proposed was used in this experiment to map several objects in the hospital room. Specifically, the Semantic SLAM algorithm provided with the tetra pack location for grasping it.

The sequence for this experiment has 6003 frames. The object models considered in this experiment were the *cabinet*, *bed* and the *tetra pack* of juice. The *tetra pack* is the same than the one in the

*https://youtu.be/kAiGgQwI684



(a)



(b)

Figure 2.7.: SLAM results at the end of the hospital room experiment. (a), top-view of the camera trajectory, estimated salient point features and recognized objects. (b), side-view of the camera trajectory and recognized objects: the tetra pack over the cabinet, both at the left; and the bed at the right.

previous experiment. The *cabinet* and *bed* are roughly modeled as delimited by planar faces. Figure 2.6 shows several frames extracted from the experiment. Figure 2.6.a (top) shows the estimation at frame #34, where already the *cabinet* has been recognized and inserted. Notice the accuracy of the insertion by the overlap between the model reprojection and the real *cabinet*. In the bottom it can be seen a top view of the 3D map: the ellipses stand for the uncertainty regions of the salient point features and the coloured prism is the *cabinet model*. As all the objects in this experiment were formed by planar faces, they are represented by coloured prisms instead of the point clouds of the previous experiment.

The following subfigures in figure 2.6 stand for other frames of the sequence temporally ordered.

Notice that in figure 2.6.c the *tetra pack* was already recognized and inserted. In figure 2.6.d the bed has also been recognized and registered. Finally, figure 2.6.e shows the final frame of the sequence along with the final estimation results (see video [†]). For better visualization, figure 2.7 shows a detail of the final map estimations and camera trajectory. Notice the accurate registration of the *tetra pack* over the cabinet in figure 2.7.b.

2.8. Discussion

State-of-the-art monocular SLAM, Structure from Motion and object recognition are combined in this work to allow the insertion of precomputed known objects into a SLAM map in addition to usual salient features. The only input to the algorithm is visual information: a monocular sequence feeds the EKF SLAM algorithm; known objects appearance and geometric models are precomputed computed from a set of sparse images; and also object recognition is driven by visual features. Experimental results show the feasibility of the algorithm and real-time capabilities for room-sized scenarios.

The approach described is the first one introducing general 3D objects in a geometric SLAM map in real-time. But we still believe that the main value of our approach resides on the concept behind it. On the one hand recent research on visual object recognition allows to recognize efficiently a wide extent of objects from visual input; but 3D information is rarely considered in those approaches. On the other hand, monocular SLAM and Structure from Motion currently offer real-time camera motion and 3D scene estimation but without a semantic meaning. The combination presented, providing a partially annotated local map and the current robot position into it, could be of high value for certain robotic tasks. Regarding the SLAM camera, only the camera calibration is needed, the camera used for building the models is different from the SLAM one. So, any robot with a calibrated camera can exploit the precomputed models in quite different scene configurations, what makes the system interoperable. The robot ends up with the location of the object it is supposed to interact with under quite general circunstantes. Finally just by recognizing an object face, the map can incorporate information about object regions not directly detected, what might be quite useful, for example in robot navigation.

Several interesting lines for future work arise from the results on this work. First, if would be very interesting to increase the quality and density of the semantic annotations. For example, the classical object recognition algorithms used in this approach could be upgraded to the most recent *category* recognition algorithms Ferrari et al. (2010). This would allow to recognize generic categories (e.g., the category chair) instead of objects, which are specific instantiations of a category (e.g., a specific chair). Context-based object detection Murphy et al. (2003) or image segmentation Gould et al. (2010) could also help to augment the density of the annotated objects. Second, monocular SLAM algorithms providing with denser geometric maps Newcombe and Davison (2010); Strasdat et al. (2010b) could be used in order to help robotic tasks like navigation or path planning.

[†] `https://youtu.be/emk1b9WTXAk`

Chapter 3

A Cloud framework for Cooperative Tracking And Mapping

3.1. Introduction

The work presented in the previous chapter shows the potential of introducing semantic information into maps. However, some algorithm used could be improved. For this reason, in this chapter we present a novelty visual SLAM algorithm based on the cloud computing paradigm.

In a practical robotic setting, the computation and memory requirements of the SLAM algorithms are two aspects of prime interest: SLAM algorithms tend to be computationally demanding and the onboard resources of a mobile robot are limited. Also, SLAM has strong real-time constraints as it is integrated in the control loop of the robot. In the latest years the possibility of massive storage and computation in Internet servers –known as *Cloud Computing* and *Cloud Storage*– has become a reality. The availability of such technology and its possible use in robotics have opened the door to a whole new line of research called *Cloud Robotics* Goldberg and Kehoe (2013). Regarding SLAM, robots could benefit from the use of the Cloud by moving part of the SLAM estimation from their limited computers to external servers; saving computational and power resources. This work tries to answer the following question *How should a SLAM system be partitioned in order to leverage the storage and computational resources in the Cloud?* Notice that the answer to this question is not trivial: Due to the real-time constraints of SLAM algorithms and the network delays the naïve solution of moving *all* the computation to the Cloud would be unfeasible. In order to guarantee the real-time, part of the computation must be performed on the robot's computers.

Our contribution is the partition of a real-time SLAM algorithm that allows to move part of the computation to the Cloud without loss of performance. Our experimental results show that the bandwidth required in all the cases does not exceed a standard wireless connection. We demonstrate the capabilities of the framework to provide the interface to a map database in a multi-map multi-camera experiment where the users can: create and save several maps, relocate in them and improve them as new areas are explored, and fuse several maps into one if an overlap is detected.

We take as starting point the monocular SLAM algorithm described in Klein and Murray (2007), the so-called *Parallel Tracking and Mapping* or *PTAM*. The processing of *PTAM* is based on two parallel threads. On the one hand, a geometric map is computed by non-linear optimization over a set of selected keyframes usually known as Bundle Adjustment. This background process is able to produce an accurate 3D map at low frame rate. On the other hand, a foreground tracking process is able

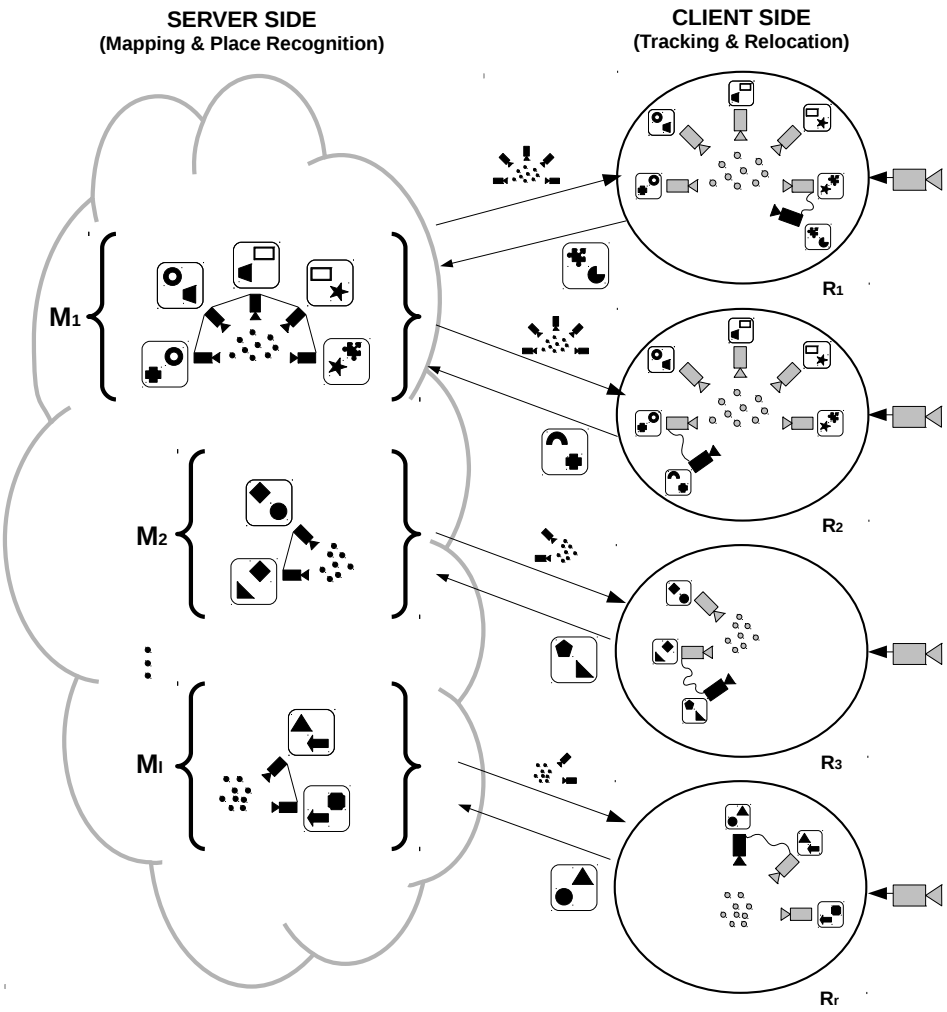


Figure 3.1.: Computer intensive bundle adjustment is performed as a cloud service running on a high performance server. Camera location with respect to the map is computed in low performance mobile devices. Several tracking threads can be run on the same map data. The data flow from tracking to mapping are the new keyframes when gathered images contain new information with respect to the available map; and from mapping to tracking is the computed map.

to estimate the camera location at frame rate assuming a known map. This method is able to produce maps composed of thousand of points using standard computers for room-sized environments. From this SLAM system, we propose C^2TAM standing for *Cloud framework for Cooperative Tracking And Mapping* that moves the non-linear map optimization thread to a service operating in the Cloud.

On top of the computational advances of the keyframe based methods, the resulting communication between the tracking and mapping processes requires low bandwidth. The tracking process sends a new raw keyframe only when the gathered image contains new information with respect to the available map. The mapping sends a new map after every iteration of the Bundle Adjustment, at a frequency substantially lower than framerate. Even in exploratory trajectories, the number of new keyframes is small compared with the frame rate; and in the case of already visited areas no new keyframes are sent. See fig. 3.1 for a scheme of the framework. The low communication bandwidth

allows to use a standard wireless connection and run tracking and mapping on different computers. Also, in both data flows the algorithm is robust to latencies. The camera tracking thread can run on suboptimal maps until the next global optimization is finished and sent. Also, with an appropriate policy on map management, the camera tracking is robust to delays in keyframe addition.

We believe that a SLAM system partially running on the Cloud has a wide array of benefits and potential applications:

- Allocating the expensive map optimization process out of the robot platform allows to significantly reduce the onboard computational budget, hence reducing the payload and power consumption; both critical factors for field robotics (e.g., unmanned aerial Achtelik et al. (2012) or underwater vehicles). More importantly, it provides the foundations to accommodate SLAM algorithms within the distributed computation framework, what makes possible exploiting the newly available Cloud computation resources.

- The interoperability between different visual sensors comes as a prerequisite in our system, as very different robots with different cameras could connect to the mapping service.

- The raw keyframe images are stored in the Cloud along with the point-based map. Optimizing a sparse 3D scene of salient points is just one of the Cloud services that can be run over the keyframes. Additionally other background processes, at different time scales, can handle map management operations aiming at life-long mapping Churchill and Newman (2012), semantic mapping Civera et al. (2011); Salas and Montiel (2011), layout estimation Hoiem et al. (2007) or computing free space for navigation Hornung et al. (2013).

- The centralized map building also allows a straightforward massive data storage of robotic sequences and geometric estimations, that could be used to provide a significant training sample for learning. It can be noticed that the size of the computer vision datasets Russell et al. (2008) tends to be much larger than the robotics ones Lai et al. (2011). The creation of datasets with data exclusively from robots is essential to exploit the commonalities in the robotic data Torralba and Efros (2011).

- The proposed framework naturally adapts to the cooperative SLAM problem Mourikis and Roumeliotis (2006); Kim et al. (2010); where several robots have to build a joint map of the environment. The server can operate on different maps and fuse them independently of the trackers running on the robots. As the number of clients grows the server computation can be parallelized. The bandwidth required for each tracker is low enough to be provided by a standard wireless.

- The technologies involved in the proposed system are in an advanced state of maturity: the Cloud Computing and storage has already been successfully incorporated to multiple domains and the keyframe-based SLAM is one the most promising mapping methods available Strasdat et al. (2010a). Additionally the proposed framework can provide the interface to an advanced database of visual maps in the Cloud.

The rest of the chapter is organised as follows: Section 3.2 refers the related work; section 3.3 discusses the main SLAM components, section 3.4 provides the formulation of the SLAM problem and section 3.5 gives the details of the proposed system, C^2TAM. Finally, section 3.6 shows the experimental results and section 3.7 concludes and presents the lines for the future work.

3.2. Related work

In recent years, the Cloud Computing paradigm has revolutionized almost every field related to computer science Armbrust et al. (2010). The idea in a few words is that the software pieces are replaced by services provided via the Internet. In the robotics community the potential applications, the benefits and the main lines for research regarding Cloud Computing have already been foreseen Guizzo (2011); Remy and Blake (2011) and some platforms are already starting to emerge Waibel et al. (2010). Nevertheless, and with some recent exceptions referred in the next paragraph, there is still a lack of specific algorithms and specific realizations of these ideas. This work aims to contribute with a concrete realization of a SLAM algorithm operating in the Cloud and a thorough experimental testing.

In Bistry and Zhang (2010), Bistry et al. analyse how SIFT extraction and matching can be moved from a robot to several servers in the Cloud. Rogers et al. (2011) queries the Cloud service *Google Goggles* to read text in signs and uses this for loop closing detection. More related to us, Arumugam et al. (2010) implements a laser-based FastSLAM algorithm running on the Cloud by distributing the particles among several computing nodes.

Very recently, Wendel et al. (2012) presented a real-time dense 3D reconstruction that shares some similarities with our approach. Specifically, this work also builds on *PTAM* and uses a distributed architecture very similar to the one presented here. Our contribution is to address the multi-user multi-map case (Wendel et al. (2012) is a single-user single-map system). The contributions that allow our system to support multiple users and multiple maps are 1) a two-step map relocation robust to communications delays, and 2) a map fusion algorithm that operates in the Cloud server.

In Castle et al. (2008, 2011), Castle et al. introduce in *PTAM* the relocation capability in a set of multiple maps. Our relocation algorithm is built on top of this work, being our main contribution over them the two-step relocation that is able to cope with the standard network delays of Cloud Computing.

3.3. A discussion on the SLAM components

The aim of this work is to provide a splitting of the SLAM problem such that its strong real-time constraints are not affected by the network delays when Cloud Computing is used. First of all, we will start by a high-level definition of the main components of a modern visual SLAM system.

1. **Mapping.** The mapping component estimates a model of the scene –a map– from sensor data. We will denote the map model as $\mathcal{M}$; and will assume that our mapping system can have several independent maps The initial approaches to SLAM used to update the camera pose and map estimation jointly and sequentially for *every* sensor data arriving Durrant-Whyte and Bailey (2006). Nevertheless, recent research Klein and Murray (2007); Strasdat et al. (2010a) have proposed a clever partition of the problem into one frame-rate thread for camera pose estimation and a second one at lower rate for map estimation; and have shown that it has computational advantages without a loss of performance or accuracy. This is hence the approach we will take in this work.

2. **Tracking.** The tracking component estimates the camera pose $\mathcal{T}^t$ for every time step t given an estimation of the map $\mathcal{M}$. A multiple-user-multiple-map SLAM system has a tracking component per user. As this pose estimation is based on the tracking of visual features in an image sequence, it should be done at a high frequency and with *strong real-time constraints*. If real-time is lost, the image tracking is likely to fail and hence the pose tracking.

3. **Relocation.** Once the tracking component has failed the relocation component tries to relocate the camera and re-start the tracking. The tracking failure can be caused by several reasons: occlusions, high-acceleration motion, blur or lack of visual features. This component also has strong real-time constraints: If the relocation takes too much time the camera might have moved from the relocation position and the tracking component might not be able to start.

4. **Place Recognition.** We understand by place recognition the capability of a SLAM system to relocate in a large number of maps. This problem is also known in the robotics community as the *kidnapped robot problem*; where a robot perceives an unknown environment and has to recognice the place it is in from a number of possibilities. Notice that the difference against relocation is the a priori knowledge on the location: Relocation comes just after a tracking failure, so we can assume a small camera motion and check close places to re-start the tracking. Place recognition does not assume any location, so every possible location is equally likely and every map in the database should be checked.

5. **Map Fusion.** Map fusion merges two independent maps into one when an overlapping area is detected by place recognition. First, we search for correspondences between local features in the two independent maps. After that, using the geometric constraints of the corresponding points, the rigid transformation between the two maps is computed. One of the maps is then put in the reference frame of the second one and duplicated points are deleted.

3.4. The SLAM formulation as Tracking and Mapping

3.4.1. Mapping

The mapping component contains l local maps $\{\mathcal{M}_1,\ldots,\mathcal{M}_k,\ldots,\mathcal{M}_l\}$. Each local map $\mathcal{M}$ is composed of a set of n 3D points $\{\mathcal{P}_1,\ldots,\mathcal{P}_i,\ldots,\mathcal{P}_n\}$ and m keyframes $\{\mathcal{C}_1,\ldots,\mathcal{C}_j,\ldots,\mathcal{C}_m\}$

$$\mathcal{M} \;=\; \{\mathcal{P}_1,\ldots,\mathcal{P}_i,\ldots,\mathcal{P}_n,\mathcal{C}_1,\ldots,\mathcal{C}_j,\ldots,\mathcal{C}_m\}\;. \tag{3.1}$$

Each map entity is modeled with a set of geometric parameters and, for most of them, an appearance descriptor. The model for a 3D point $\mathcal{P}=\{\mathbf{P},d_p\}$ contains its Euclidean 3D position $\mathbf{P}=\left(X^W Y^W Z^W\right)^\top$ and a normal $\mathbf{n}=\left(n_x^W n_y^W n_z^W\right)^\top$ in a world reference frame W; and its descriptor $d_p=\{w_1,\ldots,w_r\}$ is composed of r different patches extracted at different scales from a source image. For efficiency reasons, at the implementation level we save the pyramid at r different scales of the source image. Hence, each point $\mathcal{P}$ contains a pointer $(u\ v\ r)^\top$ to a pixel and a scale of the pyramid where the descriptor can be extracted.

A keyframe $\mathcal{C}=\{\mathbf{C},d_c\}$ is modeled quite similarly: First the 3D camera pose $\mathbf{C}=\left(X^W Y^W Z^W \alpha^W \beta^W \gamma^W\right)^\top$ using its Euclidean coordinates $\left(X^W Y^W Z^W\right)$ and roll-pitch-yaw angles $\left(\alpha^W \beta^W \gamma^W\right)$ all of them in a world reference frame W. As the keyframe descriptor d_c we use –as Klein and Murray (2008)– the frame I_c; subsampled to size 40×30, filtered with a Gaussian mask $g(\sigma)$ and normalized by substracting the mean.

$$d_c \;=\; I_c^{40\times 30} * g(\sigma) - \overline{I_c^{40\times 30} * g(\sigma)}\;. \tag{3.2}$$

For each point $\mathcal{P}_i$ we have several image measurements in different keyframes $\mathcal{C}_j$ that we will denote as $\mathbf{z}_i^j$. Each image measurement $\mathbf{z}_i^j$ puts a geometric constraint between the geometric parameters of the point $\mathbf{P}_i$ and the keyframe $\mathbf{C}_j$ given by the projection model $\mathbf{f}$

$$\mathbf{z}_i^j = \mathbf{f}(\mathbf{P}_i, \mathbf{C}_j, \mathbf{K}_j) ; \tag{3.3}$$

where $\mathbf{K}_j$ is the internal calibration of the j^{th} keyframe.

The mapping component computes the Maximum Likelihood Estimation (MLE) of the geometric map parameters $\left(\hat{\mathbf{P}}_i, \hat{\mathbf{C}}_j\right)^\top$ by minimising a robust cost function of the error $\Delta\mathbf{z}_i^j$; following what it is usually called as Bundle Adjustment Triggs et al. (2000).

$$\left(\hat{\mathbf{P}}_i, \hat{\mathbf{C}}_j\right)^\top = \underset{\mathbf{P}_i, \mathbf{C}_j}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{m} \rho(\Delta\mathbf{z}_i^j / \sigma) . \tag{3.4}$$

The error $\Delta\mathbf{z}_i^j = \mathbf{z}_i^j - \mathbf{f}(\mathbf{P}_i, \mathbf{C}_j, \mathbf{K}_j)$ is the difference between the actual image measurements $\mathbf{z}_i^j$ and the projected ones $\mathbf{f}(\mathbf{P}_i, \mathbf{C}_j, \mathbf{K}_j)$. σ is a median-based estimation of the standard deviation of the measurement's noise. As in Klein and Murray (2007), in order to avoid the influence of outlier correspondences, we do not minimize the error directly but a robust function of the error. We use the Tukey's biweight function that is defined as

$$\rho(\xi) = \begin{cases} 1 - (1 - \xi^2)^3, & |\xi| <= 1 \\ 0, & \text{else} \end{cases} ; \tag{3.5}$$

The initialization of a map is one of the key aspects in any visual SLAM system Civera et al. (2008a); Gauglitz et al. (2012). In the first frames of a sequence the *SLAM* estimation is degenerate or quasi-degenerate and hence it might fail quite often. As the *PTAM* system Klein and Murray (2007) is oriented to Augmented Reality applications, it initializes the map by asking the user to perform a careful translation of the camera in a scene with a dominant plane. We believe that this initialization procedure is not suited to a general robotic application; as we cannot guarantee that the initial motion is a translation –it will be constrained by the scene– and that the scene has a dominant plane.

For the initialization of the proposed *C²TAM* map, we have used an initial multiple model filtering scheme similar to Civera et al. (2008a). Filtering schemes are less sensitive to initialization problems in image sequences than approaches based on pairwise correspondences. This initialization process is as follows: An interacting multiple model scheme (see Civera et al. (2008a) for details) is run on the robot client for the first frames of the sequence. Once enough parallax has been detected and the estimation is not degenerate, the first frame of the sequence is set as the first keyframe $\mathcal{C}_1$ and the current frame as the second one $\mathcal{C}_2$. Both keyframes are sent to the mapping component and the optimization described above is started.

All the experiments in section 3.6 were run using this initialization. While the initialization process will have an extra computation cost on the client side; it will not be high as the map is just started and its size is small. See the details in section 3.6.1.

3.4.2. Tracking

The tracking component models each frame I^t from the image sequence as a set of geometric and appearance parameters $\mathcal{T}^t = \{\mathbf{T}, d_t\}$. The geometric parameters are those of the camera pose that

acquired the frame $\mathbf{T} = \left(X^W Y^W Z^W \alpha^W \beta^W \gamma^W\right)^\top$. The frame descriptor is composed by a global descriptor d_t^G and a set of b local descriptors d_t^L: $d_t = \left\{d_t^G, d_t^{L_1}, \ldots, d_t^{L_b}\right\}$. The global descriptor d_t^G is a subsampled, filtered and normalized version of the frame. The local descriptors d_t^L are the image patches surrounding a set of salient FAST features Rosten and Drummond (2006) extracted at 4 scales. Again, for efficiency reasons, only the image pyramid and the FAST feature positions are stored instead of the image patches.

The tracking component estimates the camera pose parameters $\mathbf{T}^t = \left(X^W Y^W Z^W \alpha^W \beta^W \gamma^W\right)^\top$ at every time step k from the information of previous camera poses, the current frame I^t and the current map $\mathcal{M}_k$. This estimation is done in 3 steps: First, the camera pose $\mathbf{T}^{t|t-1}$ at time t is predicted from the information up to the previous frame at $t-1$ applying a constant velocity model.

In the second step, the points in map $\mathcal{M}_k$ extracted at the coarsest scale –that we will name as $\mathcal{P}_k^*$– are projected into the current image I^t. For each point $\mathcal{P}_{k,i}^*$ we search for its correspondence among the closest salient points in I^t. The camera pose is roughly estimated from this first set of correspondences by a robust minimization.

$$\hat{\mathbf{T}}^{t*} = \arg\min_{\mathbf{T}^t} \sum_i \rho(\Delta \mathbf{z}_i^* / \sigma^*) \; ; \tag{3.6}$$

where $\rho(\xi)$ is again the Tukey's biweight function (equation 3.5), σ^* is a median-based estimation of the standard deviation of $\Delta\mathbf{z}_i^*$, and $\Delta\mathbf{z}_i^*$ is the reprojection error of each point $\mathcal{P}_{k,i}^*$ in the current frame $\mathcal{T}^t$ at time step t

$$\Delta \mathbf{z}_i^* = \mathbf{z}_i^* - \mathbf{f}(\mathbf{P}_{k,i}^*, \mathbf{T}^t, \mathbf{K}_t) \; . \tag{3.7}$$

Using this first estimation $\hat{\mathbf{T}}^{t*}$ as a seed, a fine grain estimation for the pose $\hat{\mathbf{T}}^t$ is finally obtained by projecting *every* map point –at every scale and not just the coarsest one– into the current frame $\mathcal{T}^t$ and minimising the reprojection error.

$$\hat{\mathbf{T}}^t = \arg\min_{\mathbf{T}^t} \sum_i \rho(\Delta \mathbf{z}_i / \sigma) \; . \tag{3.8}$$

Again $\rho(\xi)$ is the Tukey's biweight function, σ the median-based estimation of the standard deviation of $\Delta\mathbf{z}_i$, and $\Delta\mathbf{z}_i$ the reprojection error of each point $\mathcal{P}_{k,i}$ in the current frame $\mathcal{T}^t$.

A multiple-user PTAM-like system will have a pose tracking per user; hence the tracking component will be $\{\mathcal{T}_1, \ldots, \mathcal{T}_e, \ldots, \mathcal{T}_r\}$.

3.4.3. Relocation

Relocation, or the ability to quickly compute the pose of the camera when the tracking thread is lost, is done as a two step process: First, for each new image I^t, its global descriptor d_t is extracted as shown in equation 3.2. We extract the closest keyframe $\mathcal{C}_j$ in the current map $\mathcal{M}_k$ by computing the Euclidean distance between d_t and the global descriptors for each keyframe $d_{c1}, \ldots, d_{cj}, \ldots, d_{cm}$. Assuming that the 3D space is densely populated with keyframes, the nearest-neighbour of d_t will be a keyframe very close in the 3D space.

Using a simple global descriptor, like the reduced version of the image that we use, might seem to offer at first sight a poor representation of the image content. Nevertheless, several works have

proved the good performance of such descriptors. Torralba et al. (2008) shows the most relevant work on the use of downsampled and filtered images as descriptors for scene recognition as a very efficient alternative to more elaborated models with an unnoticeable degradation in performance. Such descriptor is called here *Tiny Image*. The key here is the dense population of the image space by growing the training set size to 80 million images. Recently, Milford (2013) has shown the same conclusion for the problem of place recognition in mobile robots. The good performance of this descriptor for place recognition in SLAM is also reported in Klein and Murray (2008). In this two latest references the key aspect is again the dense sampling of the image space: In Milford (2013) the experiments are done with an autonomous car that shows limited viewpoint differences. In Klein and Murray (2008) the amount of keyframes in the optimization is kept high to guarantee that a close match will always exist.

After that, the camera location of I^t is set as that of the keyframe $\mathcal{C}_j$ and the rotation is compensated by minimising the error between the global descriptors

$$\left(X^W_{\mathcal{T}^t} Y^W_{\mathcal{T}^t} Z^W_{\mathcal{T}^t}\right)^\top = \left(X^W_{C_j} Y^W_{C_j} Z^W_{C_j}\right)^\top \tag{3.9}$$

$$\left(\alpha^W_{\mathcal{T}^t} \beta^W_{\mathcal{T}^t} \gamma^W_{\mathcal{T}^t}\right)^\top = \underset{\left(\alpha^W_{\mathcal{T}^t} \beta^W_{\mathcal{T}^t} \gamma^W_{\mathcal{T}^t}\right)}{\arg\min} \left(d_t - \mathbf{w}(d_{cj}, \alpha^W_{\mathcal{T}^t} \beta^W_{\mathcal{T}^t} \gamma^W_{\mathcal{T}^t})\right) \; ; \tag{3.10}$$

where $\mathbf{w}(d, \alpha, \beta, \gamma)$ is the warping of the image descriptor d by a rotation given by the angles (α, β, γ). After this initial pose has been assigned, the tracking thread of section 3.4.2 is re-started. If the tracking is successful the camera is relocated, if not the relocation algorithm of this section is repeated again with the next image I_{t+1}.

3.4.4. Place recognition and ego-location

By place recognition we understand the ability of recognizing a part of a map $\mathcal{M}_k$ from the visual information in a frame I^t. The subtle difference with the relocation described in section 3.4.3 is the scale of the problem. The relocation component starts when the camera tracking is lost; hence we can assume that the camera has not moved much and we are in the surroundings of the latest camera pose. Only the closest keyframes in the current map $\mathcal{M}_k$ will be analysed for similarities. The place recognition component, from the information in the frame I^t, tries to recognize a map from all the maps in the SLAM database $\{\mathcal{M}_1, \ldots, \mathcal{M}_k, \ldots, \mathcal{M}_l\}$.

For place recognition and ego-location we will use then the same algorithm as in previous section; but the fact that the computation is larger will introduce differences in our C^2TAM algorithm, as is detailed in section 3.5.4.

3.4.5. Map fusion

Suppose that, from the l local maps in the Cloud server $\{\mathcal{M}_1, \ldots, \mathcal{M}_k, \ldots, \mathcal{M}_q, \ldots, \mathcal{M}_l\}$, the place recognition component from section 3.4.4 has detected that maps $\mathcal{M}_k$ and $\mathcal{M}_q$ overlap in some specific region. The map fusion components merges the two maps in a common reference frame.

Our map fusion algorithm works as follows. When the map optimization over $\mathcal{M}_k$ receives a new keyframe $\mathcal{C}^k_j$ from the tracking node; the latest is compared with every keyframe in the rest of the maps in the server. Similarly to the relocation component of section 3.4.3, we will use the global descriptor d_t as defined in equation 3.2 to quickly extract a set of potential keyframe candidates that are imaging the same area.

In a second step, once we have two potentially overlapping keyframes $\mathcal{C}_j^k$ and $\mathcal{C}_h^q$ from the maps $\mathcal{M}_k$ and $\mathcal{M}_q$ we search for point correspondences between the two maps. We project the 3D points from the two maps $\mathcal{P}^k$ and $\mathcal{P}^q$ in the common keyframe $\mathcal{C}_j^k$, resulting in two sets of image points $\mathbf{z}^{j,k} = (\mathbf{z}_1^{j,k}, \ldots, \mathbf{z}_{i_k}^{j,k}, \ldots, \mathbf{z}_{n_k}^{j,k})^\top$ and $\mathbf{z}^{j,q} = \left(\mathbf{z}_1^{j,q}, \ldots, \mathbf{z}_{i_q}^{j,q}, \ldots, \mathbf{z}_{n_q}^{j,q}\right)^\top$.

$$\mathbf{z}_{i_k}^{j,k} = \mathbf{f}\left(\mathbf{P}_{i_k}^k, \mathbf{C}_{j,k}, \mathbf{K}_j\right) \tag{3.11}$$

$$\mathbf{z}_{i_q}^{j,q} = \mathbf{f}\left(\mathbf{P}_{i_q}^q, \mathbf{C}_{j,k}, \mathbf{K}_j\right) . \tag{3.12}$$

As the two keyframes are assumed to be very similar, correspondences between $\mathbf{z}_{i_k}^{j,k}$ and $\mathbf{z}_{i_q}^{j,q}$ are computed based on their distance in the image plane $\|\mathbf{z}_{i_k}^{j,k} - \mathbf{z}_{i_q}^{j,q}\|$: If this distance is lower than a threshold (2 pixels in our experiments) the image points are considered to match. Using the correspondences between $\mathbf{P}^q$ and $\mathbf{z}^{j,k}$ –3D points in map $\mathcal{M}_q$ and image projections from map $\mathcal{M}_k$– we can compute the relative transformation between the keyframe $\mathcal{C}_j^k$ and the map $\mathcal{M}_q$ using the Perspective-n-Point (PnP) Moreno-Noguer et al. (2007). The relative transformation between the maps $\mathcal{M}_k$ and $\mathcal{M}_q$ are calculated from composition.

Finally, the duplicated points (correspondences between $\mathbf{z}^{j,k}$ and $\mathbf{z}^{j,q}$) are deleted and the rest of the points and cameras in map $\mathcal{M}_q$ are transformed according to the relative motion between the two maps and merged into $\mathcal{M}_k$.

It should be noticed that the formulation of the proposed mapping and tracking components in sections 3.4.1 and 3.4.2 uses only the *RGB* information; and hence the maps in the database are estimated up to a scale factor. This fact becomes relevant for map fusion, as two maps of different scales cannot be fused directly. There are two possible solutions. The first one is to estimate the transformation and the scale when two maps are fused; as for example in Clemente et al. (2007). The second one is, if a *RGB-D* sensor is used, to extract an estimation of the real scale of the map from the depth channel of the camera.

We chose the second option for this work: For all the points $\{\mathcal{P}_1, \ldots, \mathcal{P}_i, \ldots, \mathcal{P}_n\}$ in a map $\mathcal{M}$ we extracted their real depth measurement $D^{RGB-D} = \left(D_1^{RGB-D}, \ldots, D_i^{RGB-D}, \ldots, D_n^{RGB-D}\right)$ from the *RGB-D* keyframe where they were initialised. We then extracted their depth values at the map scale as the distances $D^{\mathcal{M}} = \left(D_1^{\mathcal{M}}, \ldots, D_i^{\mathcal{M}}, \ldots, D_n^{\mathcal{M}}\right)$ between each point position $\mathbf{P}_i$ and the position of the keyframe $\mathcal{C}_j$ where it was initialised. Finally we estimated the scale ratio as the median value of $D^{RGB-D} - D^{\mathcal{M}}$. With this value, each map can be transformed into a real-scale map and it can be fused with any other map in the database using the algorithm described in this section.

Notice that the relocation in section 3.4.3 and the map fusion in this section allow several working modes:

- Single-user-multiple-maps, where a single user is estimating a map that can be fused with other map instances in the database –previously estimated by other users. As a result, the final map comes from the fusion of two or more maps estimated at different times possibly by different users.

- Multiple-user-multiple-maps, where several users are estimating independent maps at the same time that might be fused if they belong to the same environment. As a result, we obtain a global map from several individual maps that are being estimated at the same time. After the maps are fused, the different users can keep tracking and improving the global map cooperatively.

3.5. C^{2}TAM: A SLAM in the Cloud

3.5.1. Mapping as a Cloud Service

The estimation of the map database of our system $\{\mathcal{M}_1, \ldots, \mathcal{M}_k, \ldots, \mathcal{M}_l\}$ is the most demanding computation in our framework and does not have strong real-time constraint: The map optimization in equation 3.4 can take several frames of the sequence and the tracking can still operate in a non-optimal map from a previous optimization. The mapping component might be a perfect candidate for Cloud Computing as it can tolerate the network delays; but it is also necessary that the data flow with the onboard robot computers is low enough. We will analyse that in the next sections.

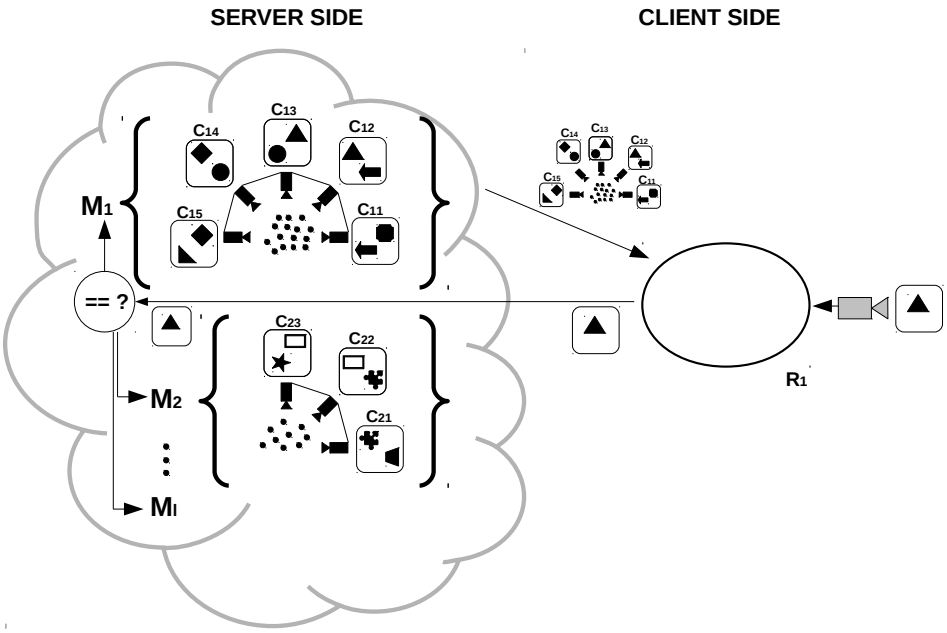


Figure 3.2.: Coarse grain place recognition in the Cloud server. The robot client $\mathcal{R}_l$ sends a frame from the sequence to the server; that tries to relocalize the camera with respect to every keyframe in every map in the database using the algorithm in section 3.4.3. If there is a match a set of close keyframes is downloaded to the client for a fine grain place recognition.

3.5.2. Tracking as a client in the robot

The camera pose tracking is a process with strong real-time constraints; that has to operate at frame rate and might fail if a few frames are skipped. This component is hence not resilient to network delays and should be allocated in the robot.

The mapping service in the Cloud receives as input new keyframes from the tracking client. This produces a low-bandwidth traffic, as typically the ratio keyframes over total frames in the sequence is quite low (in our experiments, this ratio is around 10^{-2}). The mapping serves to the client the current map $\mathcal{M}_k$ every time the map is optimized. This produces a quite high traffic

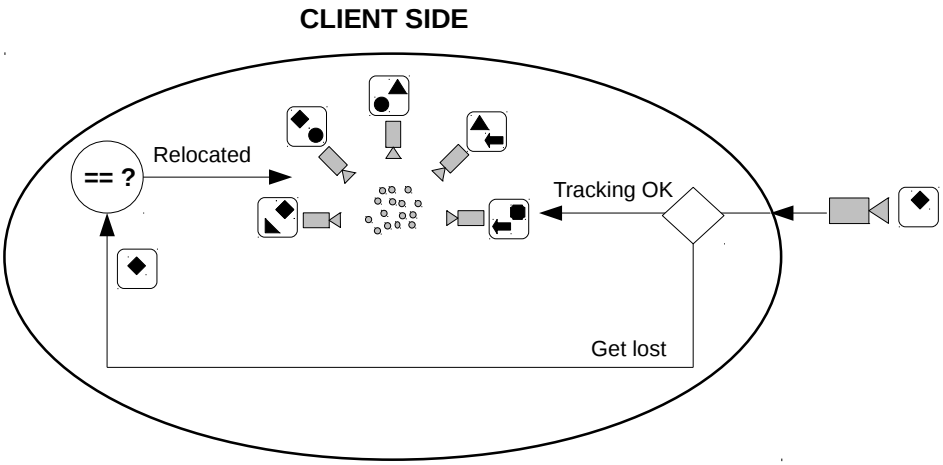


Figure 3.3.: Fine grain place recognition on the robot client. The robot client tries to relocate with respect to the set of candidates coming for the first stage, using the algorithm from section 3.4.3

3.5.3. Relocation as a client in the robot

Relocation refers to the ability of a SLAM system to relocate in a map previously estimated and stored in the Cloud. A tracking node may need to relocate in two cases: (i) the tracking node, operating successfully over a map, is lost because of a sudden motion or large occlusion. In this case, the camera is likely to be in the previous map, and relocation should only check the current map. This relocation is performed on the tracking node. (ii) The tracking thread has been lost for a long time, or just started. In this case, the camera could possibly be in a large number of maps. In this case, relocation should look for correspondences against a possibly very large number of stored maps. As this case will be more demanding, it should run partially on the mapping node.

3.5.4. Place recognition and ego-location in two steps

Place recognition and ego-location in a high number of maps can be computationally demanding, and hence it should be allocated in principle as a Cloud service. But on the other hand it is also a critical process sensitive to the network delays: If the ego-location estimation takes too much time the robot might have moved and be already in another place. This is why we propose a two-stage relocation algorithm; being the first part run on the Cloud and the second one in the robot client.

In the first stage the mapping server in the Cloud coarsely relocates the camera in a possibly large number of maps. This first coarse relocation can take a significant amount of time, and the camera may have moved when the relocation data arrives at the client. The server sends to the client a small number of filtered relocation candidates consisting of the closest keyframe from the first stage and several close keyframes from the same map; assuming that the camera may have moved during the relocation search in the server. Figure 3.2 illustrates this process.

In the second stage, the client runs a fine grain relocation among the filtered candidate keyframes sent by the server. Delays are not influential in this second stage, as relocation has a very small number of candidates and it is entirely done on the client without data transmission. A scheme of this fine-grain relocation can be seen in figure 3.3.

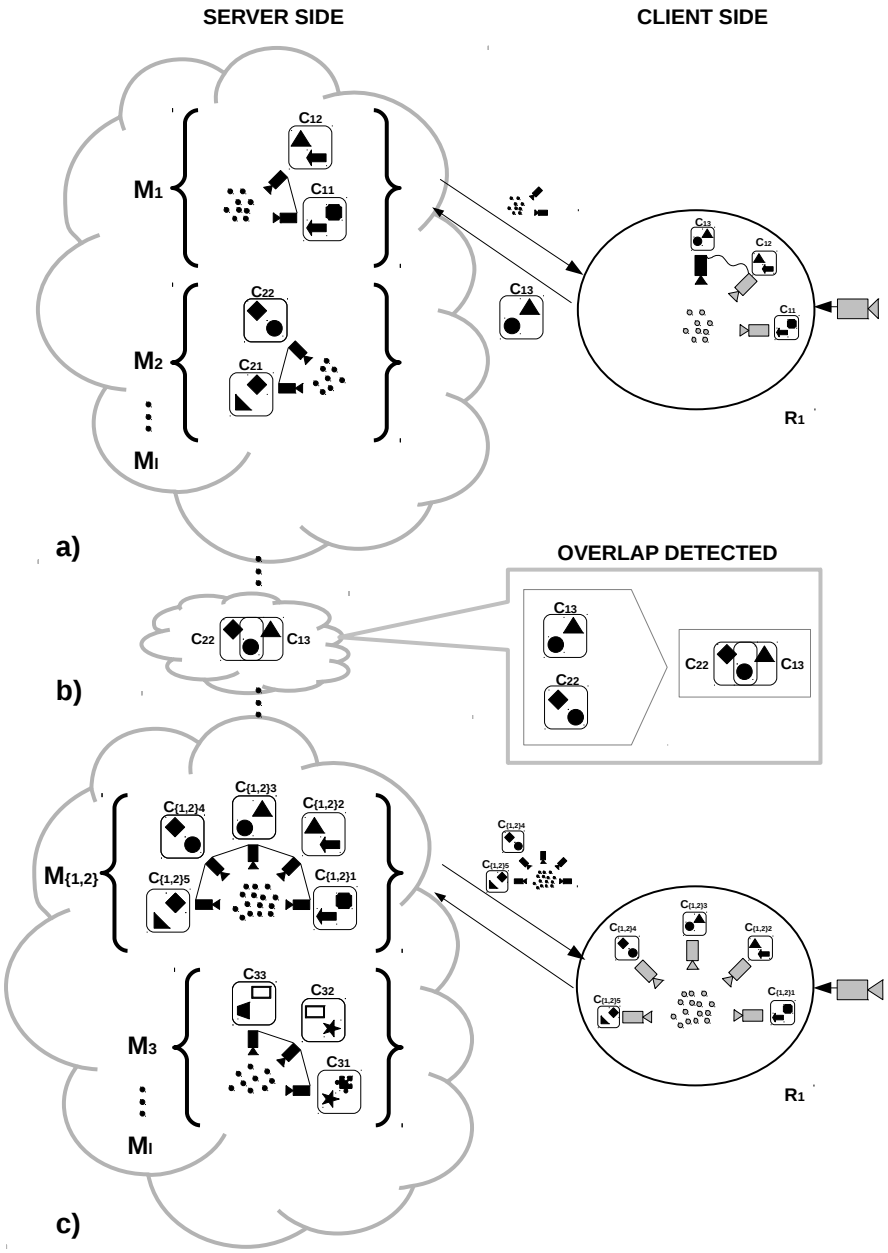


Figure 3.4.: Map fusion in the Cloud. a) The robot client $\mathcal{R}_1$ uploads a new keyframe $\mathcal{C}_3^1$ to the map $\mathcal{M}_1$. b) The keyframe $\mathcal{C}_3^1$ in $\mathcal{M}_1$ presents an overlap with the keyframe $\mathcal{C}_2^2$ in $\mathcal{M}_2$. c) $\mathcal{M}_1$ and $\mathcal{M}_2$ are fused into $\mathcal{M}_{1,2}$ and the fused map is downloaded by the robot client $\mathcal{R}_1$.

3.5.5. Map fusion as a Cloud service

The map fusion is entirely done in the Cloud server in our C^2TAM framework; as it is a process that does not have real-time constraints. An illustrative example of the map fusion algorithm in the Cloud and the associated data traffic is shown in figure 3.4. In figure 3.4.a a robot client $\mathcal{R}_1$ is tracking the camera pose and uploading a new keyframe $\mathcal{C}_3^1$ to the map $\mathcal{M}_1$. Every new keyframe that is uploaded to the Cloud server is compared with every map in the database. In this case the server detects and overlap of this new keyframe $\mathcal{C}_3^1$ in $\mathcal{M}_1$ with the keyframe $\mathcal{C}_2^2$ in $\mathcal{M}_2$. This is graphically shown in figure 3.4.b.

The map fusion algorithm from section 3.4.5 is then applied; being map $\mathcal{M}_1$ and $\mathcal{M}_2$ fused into a single map $\mathcal{M}_{1,2}$ as it is shown in figure 3.4.c. Notice in this figure that this fusion generates a high traffic between the Cloud server and the robot client; as the keyframes and points that did not have a local copy in the robot client have to be sent. But notice also that this process is not critical in time: The robot can track its pose with a suboptimal map while new keyframes are arriving, as it is shown in the experiment in section 3.6.3.

3.6. Experimental results

In this section we detail 4 experiments that show different modes of use of the proposed C^2TAM framework. All the experiments were recorded using *RGBD* cameras at 640×480. Nevertheless, it should be noticed that only the *RGB* channels are effectively used for the visual SLAM and the depth from the *D* channel is only used for visualization. All the experiments were run in real-time and using the standard wireless of our University; demonstrating that the proposed system is resilient to its network delays.

The experimental results are organized as follows: in section 3.6.1 a single-user-single-map experiment is performed to evaluate the cost and badwith required for the operation of the tracking, mapping and relocation components. Section 3.6.2 shows a single-user-multiple-maps experiment to prove the real-time place recognition capabilities. Once the real-time relocation in a stored map is shown, section 3.6.3 shows how an online estimation of a map can be fused with a previously stored map. Finally, section 3.6.4 presents a multiple-user-multiple-map experiment where two independent maps of the same room are first estimated, fused when a significant overlap is detected, and the two users keep enlarging the joint map cooperatively after the fusion.

3.6.1. Cost and Bandwidth Analysis

In this experiment, a single user is estimating a single map. Our aim is to illustrate the computational advantages of the proposed architecture with a simple example before going into more elaborated ones. The camera tracking client was run on a laptop (Intel Core i7 M 620 at 2.67GHz, 4GB RAM), the mapping service was run on a desktop PC (Intel Core i7-2600 at 3.4GHz, 8GB RAM). Both processes, tracking and mapping, have been implemented using ROS (Robot Operating System) Quigley et al. (2009) and the open source libraries of *PTAM* Klein and Murray (2007). The tracking client and the mapping server were connected through the standard wireless connection in our institution.

Figure 3.5 shows in a double-axis figure the computational cost per frame of the tracking client and the size of the map for a 4961 frames experiment. It can be seen that the tracking cost keeps being constant around $10ms$ –even when the map size is growing–, well under the threshold imposed by the frame rate of the sequence that is $33ms$. This suggests that the tracking could be done on a lower-performance computer. Notice that, with a proper policy of reducing the number of measured

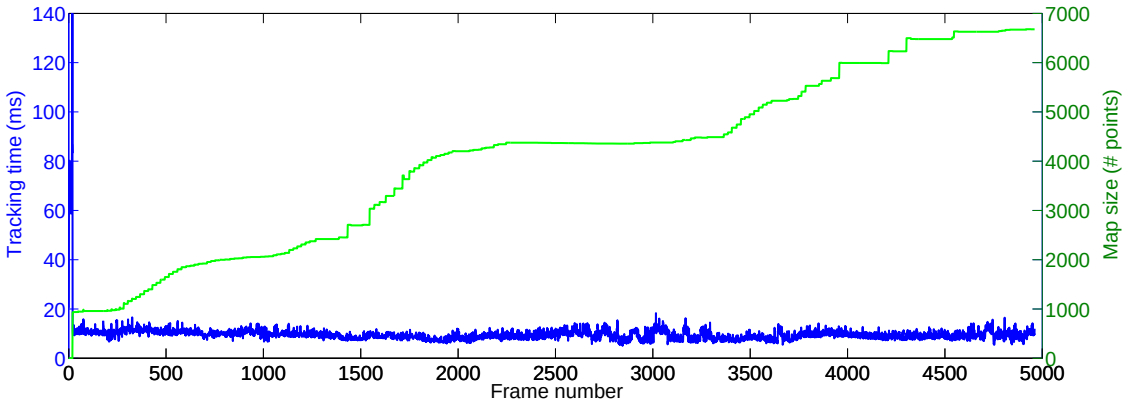


Figure 3.5.: Computational cost of the camera tracking client for a 4961 frames experiment and map size. Notice the almost constant complexity and low cost of the tracking thread, even when the map size grows.

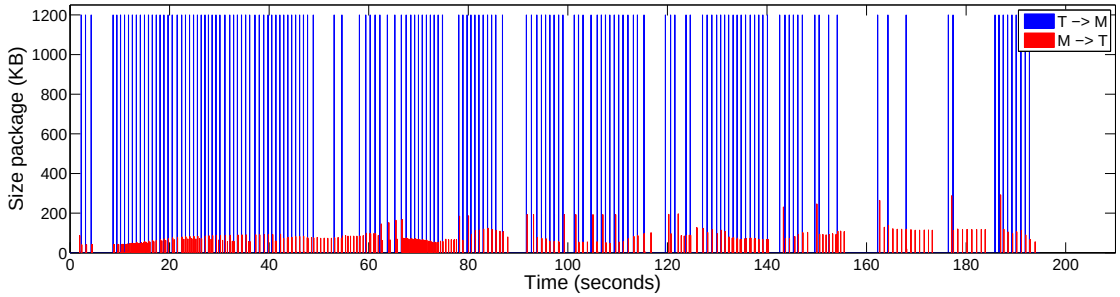


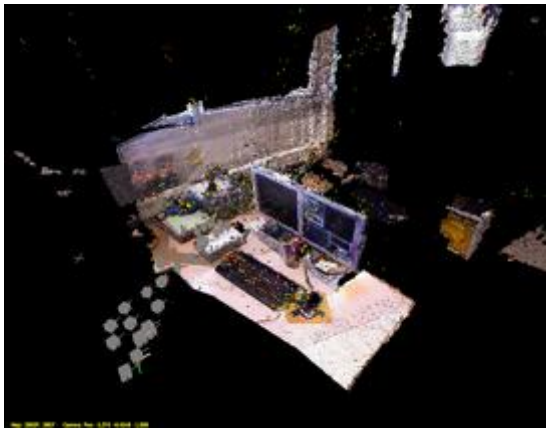
Figure 3.6.: Data flow produced by C^2TAM in a sequence of 4961 frames (around 3 minutes). Red stands for data from the mapping service to the tracking client, blue stands for data from the tracking client to the mapping server. Each peak is registered at the time the data arrives. The average data flow for this experiment has been $1MB/s$, below the usual wireless bandwidth which is $3.75MB/s$.

points –as done in Klein and Murray (2009)–, this cost could be even lower and run on mobile phone devices. Notice also the high computational cost (around $70ms$ per frame) in the first 20 frames of our algorithm, this is caused by the automatic initialization described in section 3.4.1. This is clearly a line for future work, as we are able to avoid the manual initialization of *PTAM* and the initialization algorithm works even at a lower frame rate; but the cost is rather high and should be reduced.

Figure 3.6 shows the bandwidth required for the algorithm in the same experiment. The horizontal axis is set in seconds to compare with the bandwidth of a standard wireless. The blue spikes show the required bandwidth for the data from tracking to mapping, registered at the time the data arrives. Notice that all the blue spikes are of the same height: This is because the only communication from the tracking client to the mapping server comes from uploading keyframes, which have the same size. The red spikes stand for the data flow from the mapping to the tracking. This data flow is the download of the 3D points from the map.

C^2TAM succesfully built the map from this sequence using a standard wireless connection. The average bandwidth required was around $1MB/s$; which is less than the maximum available (a usual number is $3.75MB/s$). Neither the mapping service nor the tracking client were influenced by the network latency in the communications with the proposed C^2TAM.

(a) Sample keyframes from the desktop sequence.



(b) Map estimated from the desktop sequence.

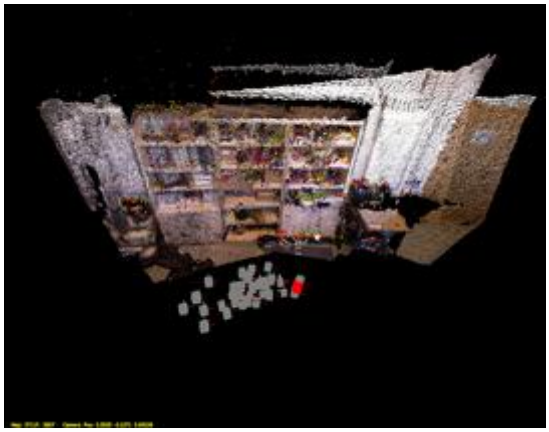
(c) Estimated map for the desktop after the laboratory sequence

Figure 3.7.: Keyframes and map for the desktop scene.

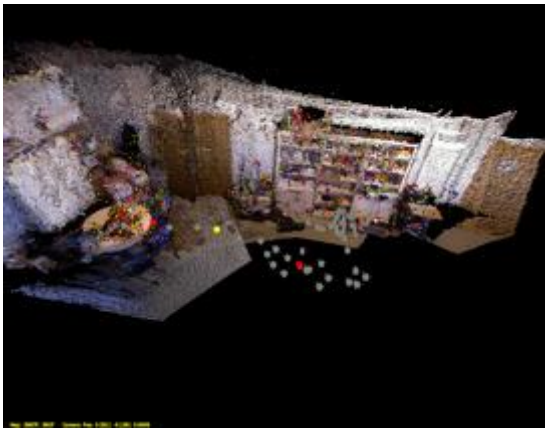
3.6.2. Relocation in Multiple Maps

In this experiment, three different maps were created using the single-user-single-map mode from previous section 3.6.1. The RGBD sequences were recorded in different areas in our research laboratory: The first one, called *desktop sequence*, was recorded in the surroundings of the desktop of one of the

(a) Sample keyframes from the wall and bookshelf sequence.



(b) Map estimated from the wall and bookshelf sequence.



(c) Estimated map for the wall and bookshelf after the laboratory sequence

Figure 3.8.: Keyframes and map for the wall and bookshelf scene.

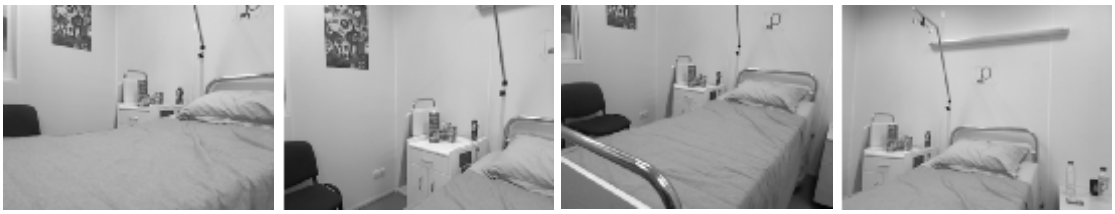
authors. The second one, called *wall and bookshelf sequence*, was recorded pointing the camera at a bookshelf. Finally, the third one is called *hospital room sequence* and was recorded in a replica of a hospital room available in our laboratory.

After each map was created, it was saved and stored in the Cloud server. Figure 3.7(a), 3.8(a) and 3.9(a) show some keyframes for each of the sequences; that is the desktop sequence, the wall and bookshelf sequence and the hospital room sequence in that order. Figures 3.7(b), 3.8(b) and 3.9(b) show a 3D view of the estimated maps of the desktop, wall and bookshelf and hospital room respectively.

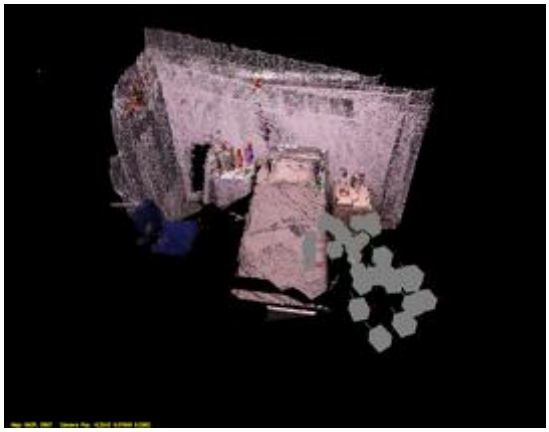
After these three maps were created, another sequence traversing the whole laboratory and hence the three above mentioned areas was recorded. Figure 3.10 shows some keyframes of this new sequence, named as *laboratory sequence*. Notice that, although this new sequence covers the same three scenes, the new keyframes show some areas that were not seen before and hence will be able to extend and improve the previous maps in figures 3.7(b), 3.8(b) and 3.9(b).

In this new sequence, C^2TAM tried to relocate the current camera in one of the three previously stored maps as described in section 3.4.3.

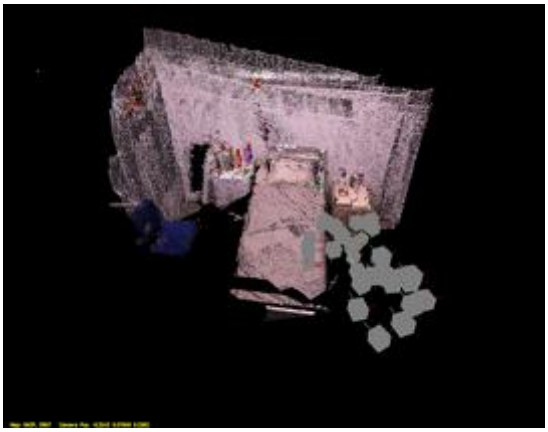
Once the camera was successfully relocated in one of the maps, the tracking thread in the client added new keyframes to this map and then extended and improved this map. Figures 3.7(c), 3.8(c) and 3.9(c) show the improved maps for each of the scenes –desktop, wall and bookshelf, hospital room– respectively. The maps were noticeably extended, as it can be seen comparing against the maps before the *laboratory sequence* was processed (figures 3.7(b), 3.8(b) and 3.9(b)). Specifically the *desktop map* was extended from 2662 points and 28 keyframes to 3313 points and 47 keyframes,



(a) Sample keyframes from the hospital room sequence.



(b) Map estimated from the hospital room sequence.



(c) Estimated map for the hospital room after the laboratory sequence

Figure 3.9.: Keyframes and map for the hospital room scene.

Figure 3.10.: Sample keyframes from the sequence traversing the whole laboratory.

the *wall and bookshelf map* from 2711 points and 32 keyframes to 3987 points and 58 keyframes, and the hospital room from 642 points and 29 keyframes to 1624 points and 58 keyframes.

After relocation and improvement process, a set of measurements have been performed in order to verify the accuracy of the maps generated by C^2TAM. Figure 3.11 shows the measurements taken, and the Table 3.1 contains the comparison of these measurements with the ground truth of the scenarios. The ground truth measurements was made by a tape measure with millimeter precision due to the tape resolution

Map	Element	C^{2}TAM	ground truth
Desktop	Laptop	0.340	0.337
Desktop	Table	1.107	1.102
Desktop	Window	1.205	1.201
Room	Bed	2.039	2.044
Room	Cabinet	0.418	0.420
Room	Walls	3.105	3.096
Wall	Bookshelf	2.723	2.727
Wall	Door	1.790	1.795
Wall	Table	1.106	1.107
Wall	Whiteboard	1.267	1.264

Table 3.1.: Comparison between *C^{2}TAM* measurements and the ground truth. All the measurements are in meters.

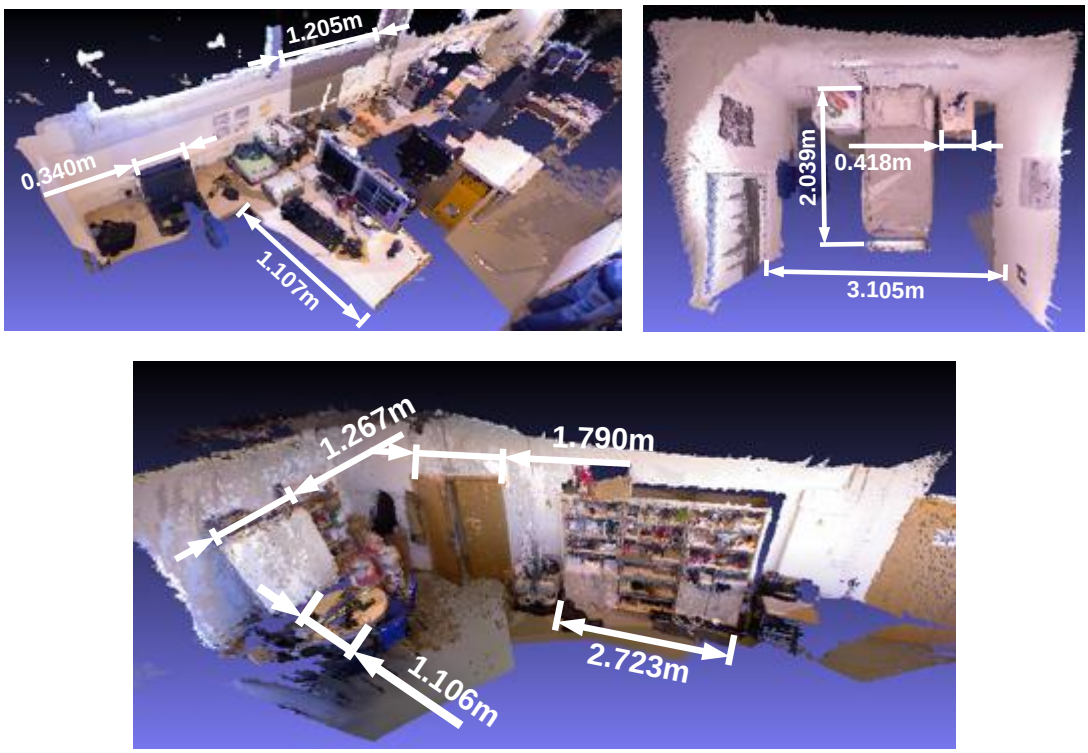


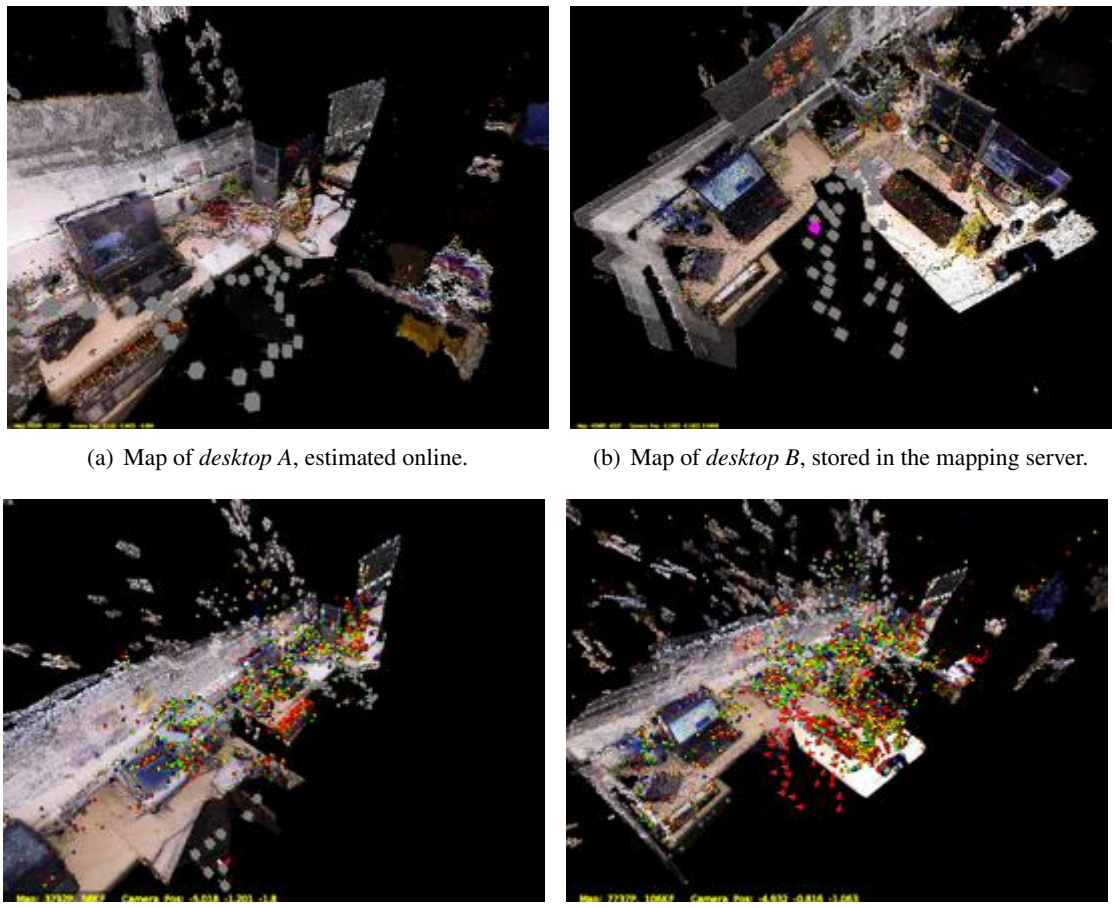
Figure 3.11.: Set of the measurements taken on the scenarios.

3.6.3. Overlapping map fusion

This experiment aims to show the map fusion capabilities of the proposed *C^{2}TAM* system. The mapping server contains the maps estimated for the previous experiments. We run the *C^{2}TAM* on a sequence imaging a desktop. We will denote this scene as *desktop A*. The camera later moves to another desktop that we will call *desktop B*. Figure 3.12(a) shows the estimated map of *desktop A*, just before moving to *desktop B*.

Among the maps in the server there is an estimated map of *desktop B*, which is shown in figure 3.12(b). As the camera goes from *desktop A* to *desktop B* and adds keyframes to the map, the similarity of the new keyframes and every keyframe in every other map in the server is computed. When the camera reaches *desktop B* the map fusion process detects the similarity with a keyframe from a previous map (see the keyframes from both maps in figure 3.13) and merge them into one. Figure 3.12(c) shows the map when the similarity is detected, and figure 3.12(d) shows the map after merging with the map in the database. Notice the correct alignment after the map fusion in this latest figure.

Regarding the map fusion process, the communication flow between tracking and mapping is the critical point and C^2TAM deals with it giving an approach that manage efficiently the amount of data exchange between processes. Once the overlap is detected the mapper server merges both maps and has to send the new keyframes and points to the tracker.



(a) Map of *desktop A*, estimated online.

(b) Map of *desktop B*, stored in the mapping server.

(c) Map of *desktop A* when a common area with the map of *desktop B* is detected.

(d) Maps *desktop A* and *desktop B* after the fusion.

Figure 3.12.: Several snapshots of the maps for the map fusion experiment.

In the experiment proposed, the current map desktop A (112 keyframes and 7925 points) is fused with a previous map desktop B stored on the server (42 keyframes 4098 points). This process implies to send the new keframes and point from server to client. In this case the data flow generated is 16 MB, according to the standard wireless bandwidth, this communication spends arround 5 seconds. This time could be reduced taking into account the principle of locality: if the server starts to send

only the informations related with the keyframes close to the pose of the current camera, the data flow will be reduced.

Figure 3.13.: Images of the keyframes of the previous (left one) and the actual map (right one).

Figure 3.14.: Panoramic snapshot of the experimental environment room.

Despite the amount data flow exchanged, the client performance is not affected. The client is working with a suboptimal map while the mapping is sending the new information about keyframe and points. The client can work properly because the working area is covered by keyframes contained in the suboptimal map. Once new information (keyframes and points) arrives, the client takes into account this information on the tracking proccess and update the local copy of the map. For more details on the experiment in this section, the reader is referred to the video *.

3.6.4. Cooperative SLAM

This section shows a cooperative SLAM experiment using the proposed framework. The aim is mapping an office using several cameras initially unaware of each other. The clients attached to the cameras perform tracking in each map separately. As shown in section 3.6.3, the server optimizes every map but also looks for overlap between them. When the overlap is detected, the server fuses both maps. From there, both trackers operate on the new fused map.

Specifically, we used two RGBD cameras and cooperatively mapped an office in real-time. Two laptops (Intel Core i7 M 620 at 2.67GHz, 4GB RAM) are attached to the cameras and act as clients. A PC (Intel Core i7-2600 at 3.4GHz, 8GB RAM) in other building of our university acted as the server. In several parts of the experiment the trackers were lost and were able to relocalize as explained in section 3.4.3.

*https://youtu.be/kE5wmFoCV5E

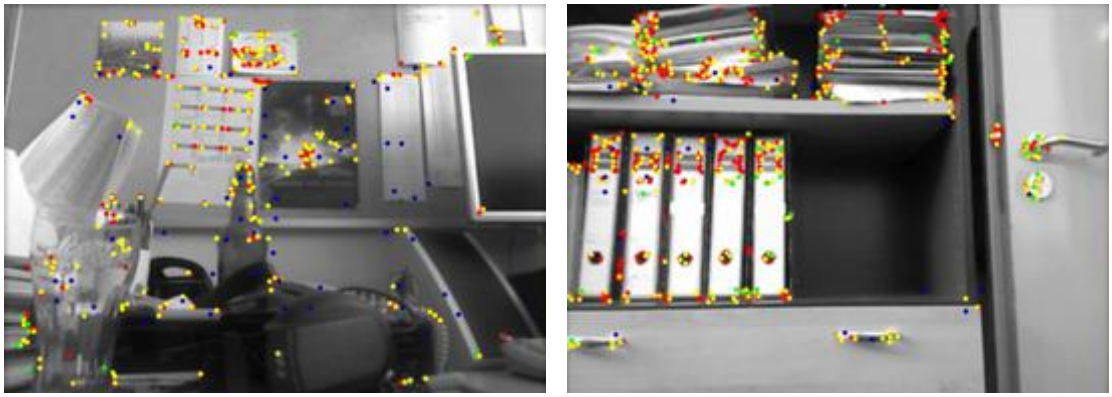


Figure 3.15.: Initial images taken by first tracker (left) and second tracker (right).

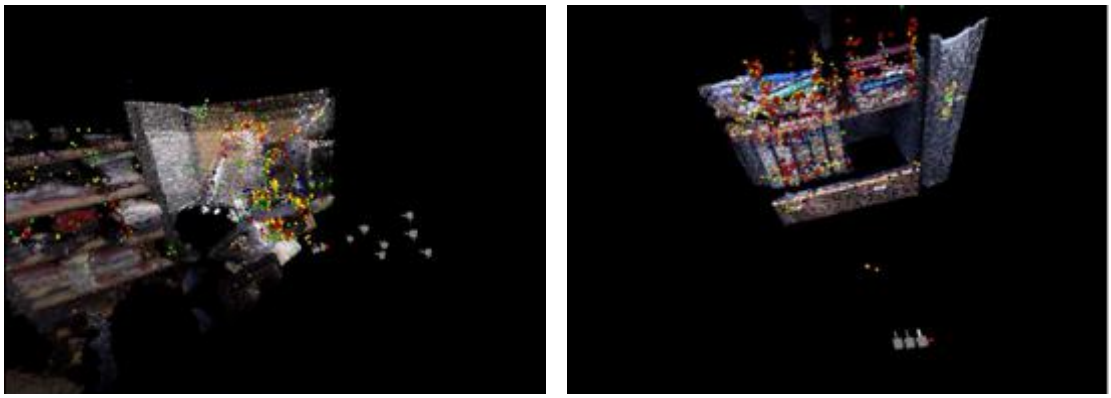


Figure 3.16.: 3D estimated maps by first tracker (left) and second tracker (right).

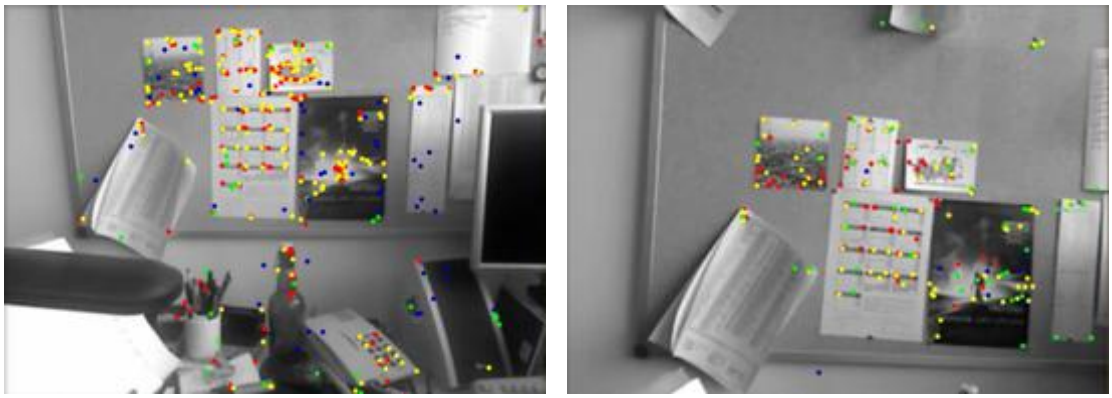


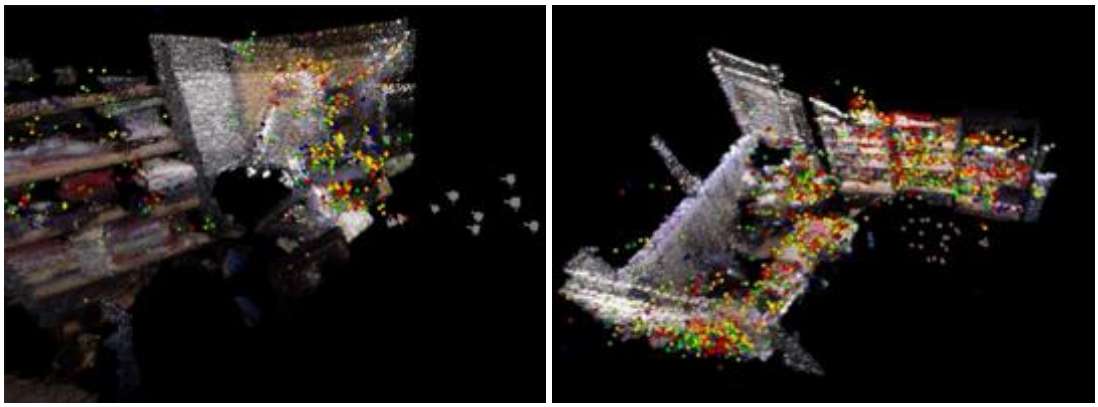
Figure 3.17.: Images of both trackers on the same area. First tracker (left one) second tracker (right one).

Figure 3.14 shows the scene to be mapped. Figure 3.15 shows the initial images taken by the two cameras. Notice that the areas do not overlap and hence two different maps are started: the first one

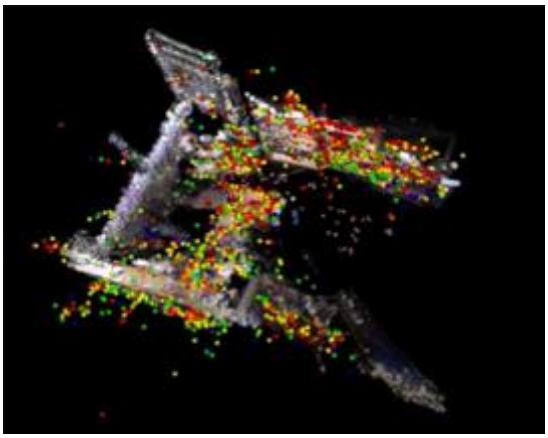
of a desktop and the second one of a bookshelf. Figure 3.16 shows the 3D estimated maps after some keyframes have been added.

When the second tracker approaches the desktop area, eventually a significant overlap is found. Figure 3.17 shows the actual image of the second tracker just before the fusion and an image of the first tracker from the beginning of the experiment when this area was mapped. Notice that the image approximately shows the area above the desktop where the first map started.

After the overlap detection and motion estimation, both maps are finally fused. Map 1 contained 25 keyframes and 1578 points, and map 2 38 keyframes and 4833 points. The fused map contains 63 keyframes and 6302 points. Notice that while the number of keyframes is the sum of the individual maps, this does not happen with the number of points as the duplicated ones are deleted. Figure 3.18(a) shows the individual 3D maps of each tracker before the fusion and figure 3.18(b) shows the final map after the fusion. After the fusion both trackers map the scene over the same map. The following figure 3.19 shows the potential of the framework for cooperative SLAM: the second tracker stops mapping (figure 3.19(b)), and the first tracker continues exploring and expanding the office map (figure 3.19(a)). Figure 3.19(c) shows the map before and after the expansion. Finally, when the tracker two starts moving again, it is able to relocalize in the area that tracker one has mapped as both are working on the same map instance.



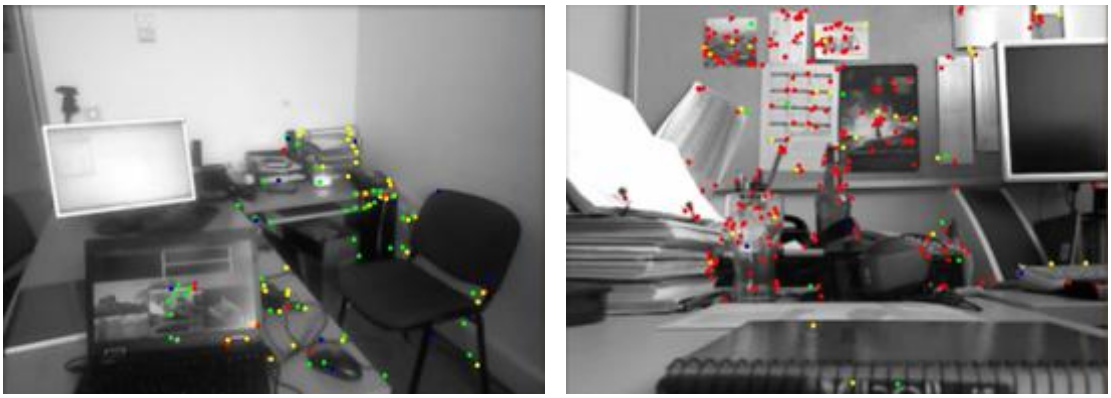
(a) 3D maps of each tracker before fusion. First tracker (left one) second tracker (right one).



(b) Final map after fusion both maps.

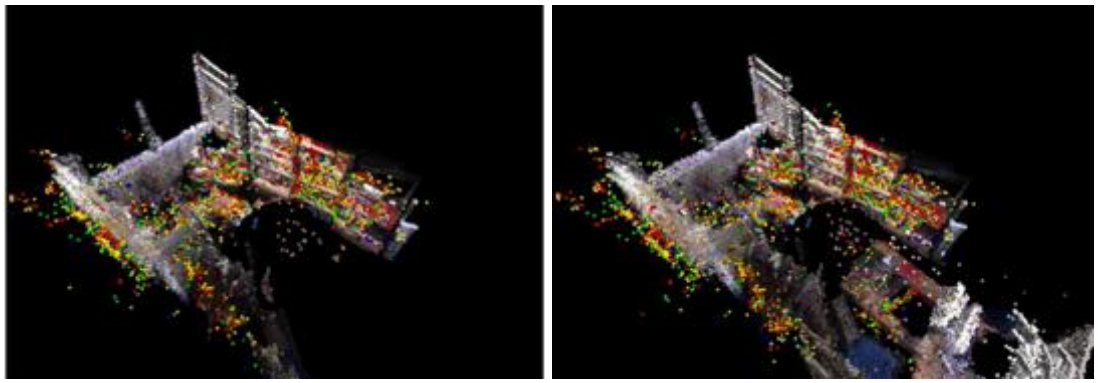
Figure 3.18.: 3D maps before and after fusion process.

Figure 3.20 shows the final 3D reconstruction build cooperatively by the two cameras. This final map contains 100 keyframes and 7483 points. We encourage the readers to see the video of the experiment [†] for further understanding.



(a) First tracker image.

(b) Second tracker image.



(c) 3D map before and after the first tracker update

Figure 3.19.: Cooperative update of the map.

3.7. Discussion

This work presents a novel framework for distributed keyframe-based SLAM where the map optimization is moved to a server or an array of them outside the robot –the Cloud– and the robot only has to run a light camera tracking and relocation clients and have access to an Internet connection. A direct consequence is that the computational load on the client side –on the robot– is reduced. This reduction might be critical in robotic applications with strict constraints on both power and weight, like unmanned underwater or aerial vehicles.

Our algorithm exploits the fact that the state-of-the-art visual SLAM algorithms divide the Simultaneous Localization and Mapping problem into two parallel threads, one for camera pose tracking and the other one for map optimization. Our experimental results demonstrate that the communication between the two threads is small enough to be supported by a standard wireless, and that the latency

[†] https://youtu.be/giMDnKhkg-0

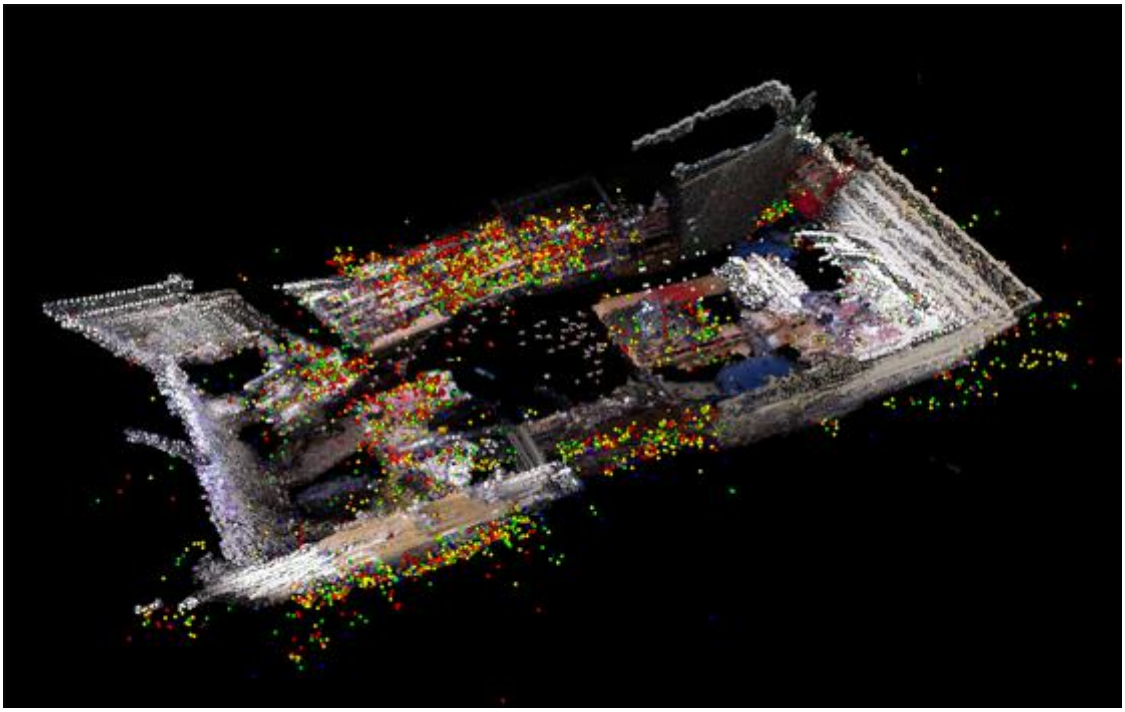


Figure 3.20.: 3D map reconstruction by the two cameras.

introduced by the network does not influence the performance of the algorithm. In our experiments we use an *RGBD* sensor, using the *RGB* images to align the cameras and the point clouds from the *D* channel for visualization and map fusion.

The most critical point of the algorithm where the latency is important is the relocation with previous maps. This approach contributes with a 2-stages relocation algorithm; an expensive coarse relocation is run on the server over all the stored maps and returns the specific map where the camera is, and a second fine relocation downloads this map to the client and runs a cheap relocation on the keyframes of this map. We have also demonstrated the ability to build maps concurrently between several sensors observing the same environment.

We have implemented a map fusion component that enables the possibility of building a cooperative map of an environment. Each robot can explore a new area and estimate a map while the Cloud server looks for similarities with the maps that the rest of the robots are estimating. Once the map of one robot is detected to have a common area with another map, the Cloud server will fuse both into a single one independently from the tracking processes.

We believe that the allocation of the bulky map estimation and management in a Cloud of servers outside the robot sets the basis for the mapping systems to exploit the next step of computing power in order to keep track of Moore's Law. Additionally it opens the door to a new array of possibilities: 1) The online estimation can be massively parallelized and hence very large maps can be optimized in a short time (like in Furukawa et al. (2010)). Also, parallelizing the relocation could boost the number of maps that can be managed in a reasonable time. 2) The developed algorithms provide the basis for the interface of a map database in the Cloud. 3) The map can be improved and enriched offline with expensive computations that cannot be done online, both geometric (e.g., map smoothing assuming planar environment Gallup et al. (2010) or free space estimation Hornung et al. (2013))

and semantic (e.g., recognition of objects Civera et al. (2011); Salas and Montiel (2011)). The new geometric or semantic features will be available if a later user relocates itself in a previous map and downloads this map. 4) The massive storage of maps in the Cloud could serve as a training database for learning algorithms to model the commonalities of robotic maps and their variations in the temporal dimension.

Chapter 4

Semantic Mapping System: A Cloud Enabled Knowledge-Based Approach

4.1. Introduction

In chapter 2 we demonstrate the importance of semantic maps through implementation of a proof of concept of a Semantic SLAM, however some point needed to be improved. Previous chapter focused on improve the SLAM algorithm, designing and implementing a new one based on a cloud computing paradigm. In this chapter we will address the improvement of the object recogniser using an algorithm based on Bag of Words and we also demonstrate how the navigation of a robot could be improved using the semantic information generated.

The aim of this works is to investigate a Web-enabled and knowledge-based approach to semantic mapping in order to build models of the environment and explore the role that cloud services can play in this mapping approach. As we introduces in chapter 3 the use of these cloud services has recently opened a new line of research in robotics called *Cloud Robotics* Kehoe et al. (2015). In Stenmark et al. (2015) different architectures based on a knowledge-based solution have been presented in industrial robotized automation systems. Salmerón-García et al. (2015) and Kehoe et al. (2015) explore the use of a Cloud Computing for offloading intensive computing tasks like vision-based algorithms and grasp planning respectively. In particular, we consider a simple robot that has access to the cloud-based RoboEarth knowledge base Waibel et al. (2010), and evaluate how access to such a cloud-based knowledge base can help robots with their tasks. RoboEarth enables robots to upload and download "action recipes", models of objects they have created and maps of environments. By intelligently selecting only those pieces of information that are needed for the current task, robots can keep their local knowledge bases and object model database small and efficient, while having much larger information resources in the background.

All pieces of information in RoboEarth are semantically annotated, i.e. they are described in a formal, logical language Tenorth et al. (2013) and are linked to an ontology. To achieve platform independence, these annotations include a specification of which capabilities a robot needs to have in order to execute a task. When searching for suitable "action recipes", a robot can match this specification against a formal model of its own components and capabilities described in the Semantic Robot Description Language (SRDL), Kunze et al. (2011). If necessary components or capabilities are missing on the robot, the recipe cannot be executed and is not considered for download. If all required capabilities are available, the robot model is used to generate a plan that is tailored to the

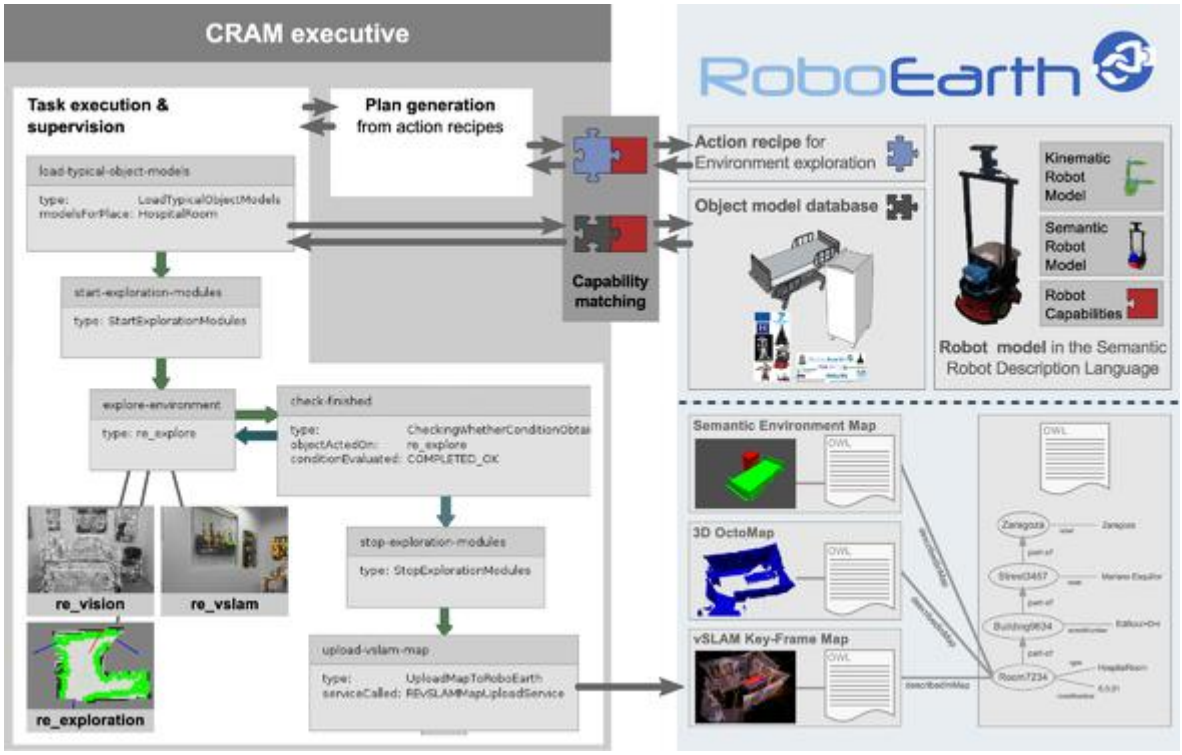


Figure 4.1.: Overview of the proposed system. In the beginning, the RoboEarth knowledge base (right) contains only the elements above the dotted line: An action recipe describing the exploration task, a set of object models and the robot's SRDL description. When a robot requests an action recipe, it is matched against its capability model and, if all required capabilities are available, a plan is generated. During execution of this plan (left part), the robot first downloads a set of object models that are to be expected in this environment and uses these models to build a semantic map. After execution, it uploads the generated set of maps to RoboEarth (lower part of the right block) to make them available to other robots.

hardware of the respective robot. The semantic annotations further enable robots to perform logical inference, for instance to decide which are the most likely objects in a room (and only download their models to their local database), or where novel objects are likely to be found (and guide the search accordingly).

In order to apply abstract knowledge to operation in the real world, it needs to be *grounded* Harnad (1990) in the robot's perception system and its knowledge about the environment. In this work, we propose to link the knowledge base with a visual SLAM system that provides accurate and continuous asynchronous perception. The system is integrated with an object recognition module that identifies objects based on a local database of object models. The main contributions of this work are (1) a semantic mapping method resulting from the synergistic integration of a visual SLAM map of objects with the RoboEarth ontology; (2) knowledge-based methods for using prior information, exemplified in the selection of object models for exploration and in the guidance of a robot when searching for a novel object; and (3) methods for embodying the semantic map building and exploitation in a simple robot using RoboEarth cloud services.

RoboEarth boosts mapping by providing: (1) a subdatabase of object models relevant for the task at hand, obtained by semantic reasoning, which improves recognition by reducing computation and the false positive rate; (2) the sharing of semantic maps between robots, and (3) software as a service to externalize in the cloud the more intensive mapping computations, while meeting the mandatory hard real time constraints of the robot.

The rest of the chapter is organised as follows: Section 4.2 discusses related work and section 4.3 presents an overview of our system. Next in section 4.4 we explain the two main tasks performed: The creation of an initial semantic map building and knowledge-guided object search. Section 4.5 presents the robot capabilities and section 4.6 summarizes the process of reasoning about object locations. Finally, section 4.7 shows experimental results and section 4.8 concludes and presents lines for future work.

4.2. Related Work

Several proposals have been made for building maps of objects. Objects from a database are recognized and located in Günther et al. (2013) where polyhedral CAD object models are recognized in single RGBD images. Similarly, using point clouds, in Mason and Marthi (2012) geometrical primitives are segmented assuming they correspond to scene objects. Combining visual SLAM with object recognition to produce maps of objects has recently gained more attention for pure visual RGB sensors in Castle et al. (2010); Civera et al. (2011), and for RGBD in Salas-Moreno et al. (2013). Several approaches have been made to endow maps with reasoning capabilities. A Bayesian network classifier is proposed in Vasudevan and Siegwart (2008) to encode the relations between objects in a scene and the objects typically present in a type of room. An ontology-based approach is proposed in Galindo et al. (2008); Pangercic et al. (2012) to represent knowledge about the elements in a map. The knowledge-based maps by Zender et al. (2008) provide grounding by combining place recognition from 2D laser maps and object recognition. An exploration method similar to ours has been proposed in Meger et al. (2008). Our contribution is to combine a knowledge-base with a visual SLAM map of objects to ground the robot perceptions to implement the RoboEarth Web and cloud mapping services. For the estimation of this semantic map, we propose the use of action recipes that describe how to explore the free space while searching for objects in a local database using an object recognition algorithm.

Structured object search and reasoning about likely object locations have been an active research topic over the past years. Much of the work has explored vision-based methods to search for objects in a top-down manner based on saliency and visual attention mechanisms Oliva et al. (2003); Ekvall and Kragic (2005); Shubina and Tsotsos (2010). Having a (partial) semantic map allows a robot to apply background knowledge for directing the search. One possibility is to learn co-occurrence statistics of object types and object–room relations, for example from online image databases Kollar and Roy (2009) or from search engine results Zhou et al. (2012). Joho et al. (2011) use co-occurrence information and other heuristics for efficiently searching for objects in structured environments, in particular supermarkets. Schuster et al. exploit similarity scores computed based on an ontology of object types for directing the search towards locations where semantically similar objects are known to be Schuster et al. (2012). Kunze et al. propose a utility-based approach for object search that particularly focuses on the decision of which location to search first Kunze et al. (2012). This work was extended in Kunze et al. (2014) to use geometric models of directional qualitative spatial relations with respect to landmark objects and to use 2D cones for approximating the sensor field of view. Wong et al. include manipulation actions into the object search, which allows the robot to reason about which objects have to be removed before being able to see the target object Wong et al. (2013).

The approach by Aydemir et al. Aydemir et al. (2013) is similar to ours in that they also use landmark objects to guide the search for smaller objects inside or on top of the landmarks. While they focus on the probabilistic formulation of the search procedure as a Markov Decision Process, we explore a knowledge-based strategy that exploits formal knowledge about object types, their (likely) spatial relations, and their shape and appearance.

4.3. System overview

Figure 4.1 shows the typical workflow of a robot using the system. We assume that the RoboEarth knowledge base (right block) contains the required task descriptions (called "action recipes") and object models. In this work, we focus on two action recipes for (a) semantic mapping of an unknown environment and (b) active search for an object based on a partial semantic map. The locations of objects already detected in the room thereby serve as landmark objects.

Each piece of information is annotated with a description of the capabilities required for making use of it (depicted as colored puzzle pieces), that is matched against a formal model of the robot's capabilities described in the SRDL. Based on the background knowledge about which objects are likely to be encountered in which kinds of rooms, RoboEarth infers a set of object models that can be recognized during the exploration.

After download, a robot plan is generated from the action recipe (section 4.4) and the task is executed accordingly. The robot explores the environment using a frontier-based algorithm (*re_explore* component), recognizes objects using the *re_vision* module and inserts them into a map build by the *re_vslam* module. After the exploration has finished, the robot exports the map in the formal RoboEarth language and uploads it to the RoboEarth knowledge base.

4.4. Action Recipes for Active Perception Tasks

Action recipes abstractly specify which actions need to be performed to accomplish a task in a (largely) robot- and environment-independent manner. RoboEarth aims at the exchange of recipes between heterogeneous robots in different environments, which therefore need to be reduced to a description of the task itself, eliminating all hardware- and environment-specific parts. While the resulting descriptions can easily be transferred to another robot, they are too abstract to be directly executable. The robot thus needs to interpret the instructions, fill in missing information, and select and parameterize suitable "skills" that provide the implementation for the respective action steps. The capability matching procedure described in section 4.5 verifies that all skills needed for executing a recipe are available on a robot.

Action recipes are formulated in the RoboEarth language Tenorth et al. (2013) that is based on the W3C-standardized Web Ontology Language OWL W3C (2009). Actions in a recipe are described as classes whose properties described action parameters such as the *objectActedOn*. They can inherit properties from more generic classes in the knowledge base, which we often use for inheriting information about required capabilities. This way, the recipes can be kept short and concise, since 'common-sense' knowledge does not have to be communicated. Examples of action recipes for exploring an environment and searching for objects can be found in figures 4.1 and 4.3, respectively. These recipes can easily be created using a graphical editor without knowledge of the OWL language.

As mentioned earlier, action recipes are not executable by themselves, but aggregate "skills" (that implement single action steps) into more complex task structures. Our system uses the Cognitive Robot Abstract Machines (CRAM) executive Beetz et al. (2010) for controlling the robot, so the

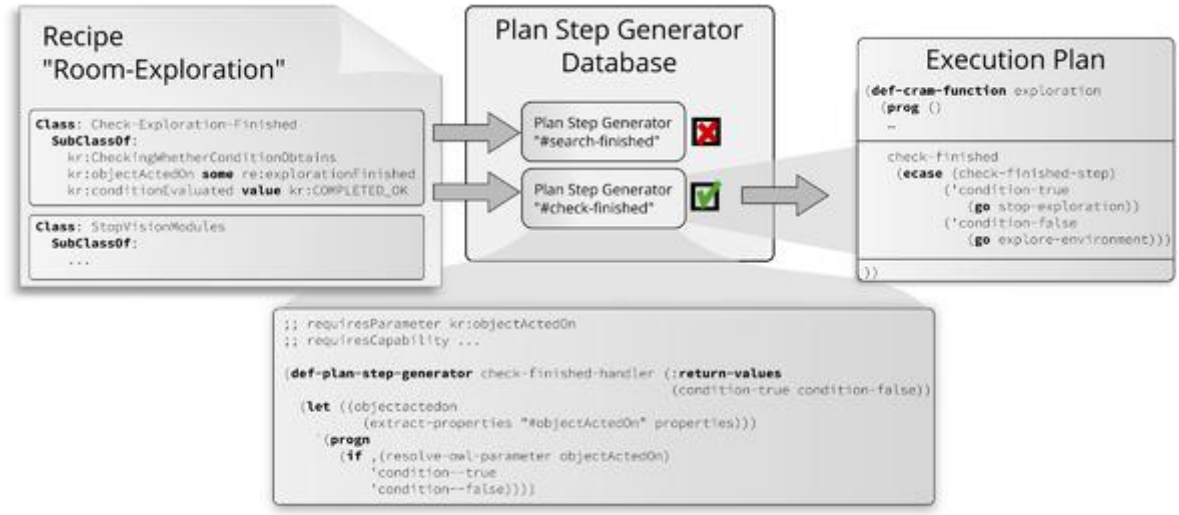


Figure 4.2.: Generation of the execution plan. The recipe (left) is an OWL document composed of parametrized subactions, described in terms of OWL classes. To generate the plan, the system looks in the database for code generating functions that are applicable on the specific instance and robots (bottom), and inserts the resulting function into the final execution plan (right).

"skills" correspond to fragments of the robot plans. These fragments are not static, but are generated by Lisp macros that are parameterized with the OWL description of an action step (figure 4.2). This allows to consider the action context as well as the robot model to generate tailored plans. For example, when generating the code for computing object visibility (sec. 4.6), the pose of the camera relative to the robot base is read from the robot's SRDL description.

The code generation macros are also stored in the RoboEarth database and can therefore be shared among robots. For each action described in the recipe, the system searches for suitable macros considering the robot's capabilities. In case multiple results are found, the one with the minimal semantic distance (estimated via the Rada distance Blanchard et al. (2005)) to the action at hand is selected. If the result is still ambiguous, a human operator is asked. The code generation macros then extract the required action parameters from the robot model and the semantic environment map.

As part of this work, we have created two action recipes to enable a simple robot to perform semantic mapping in the cloud using RoboEarth. The first action recipe (fig. 4.1 left), sketched in Algorithm 1, enables a robot to build a semantic map for a novel environment, exploiting prior information about the room type. The second one (fig. 4.3) illustrates how information from the semantic map can be exploited when searching for an specific object. Algorithm 2 sketches the steps of this recipe. The described recipes build upon a set of perception and navigation capabilities that are detailed in section 4.5.

4.4.1. SemanticMapping Action Recipe

The execution of the *SemanticMapping* action recipe results in an exploratory behavior of the robot. Before starting the exploration, the knowledge base infers a set of landmark objects that are typically found in the type of room to be explored. These models are loaded into a local subdatabase on the

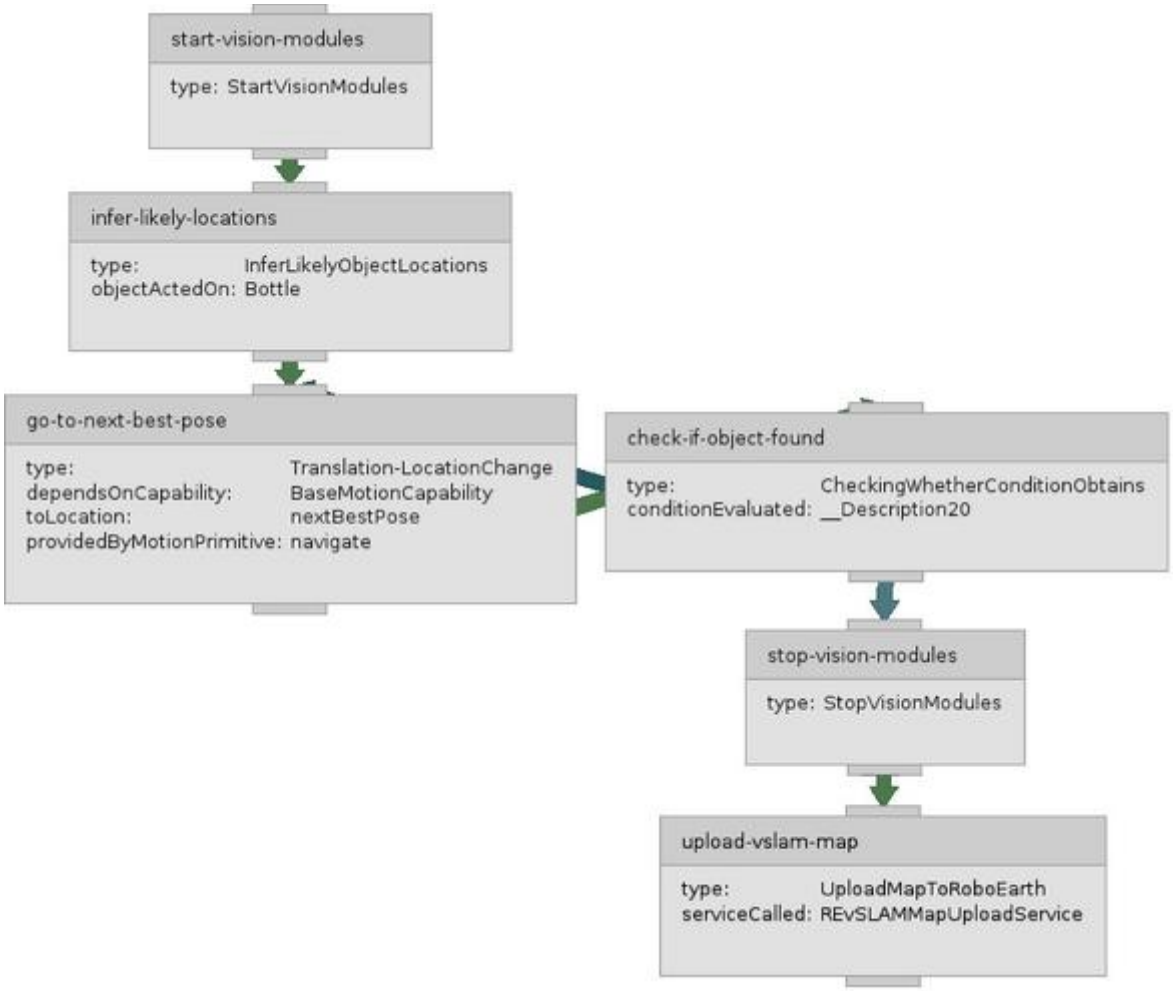


Figure 4.3.: ObjectSearch Action recipe task execution

robot that allows real-time object recognition for map building. It further increases the recognition precision and recall because only objects that are likely to be in the room are searched for. After completing the room exploration, it produces a semantic map that is stored in the RoboEarth knowledge base.

The recipe commands the robot to explore the room while avoiding obstacles. Simultaneous to room exploration, the visual SLAM builds a map providing locations for selected geometrical features and landmark objects recognized in the scene. Once the exploration is finished, the object instances are linked with the RoboEarth ontology in order to upgrade the map of objects into a semantic one. The semantic map along with the occupancy grid and features map are uploaded as a RoboEarth environment.

4.4.2. ObjectSearch Action Recipe

The *ObjectSearch* recipe assumes that a (partial) semantic map valid for the room is already stored on a RoboEarth environment. Based on the locations of landmark objects in this map, the knowledge base

infers potential locations from where the object might be detected. From the occupancy map, the free space for robot navigation is computed, and according to the robot's SRDL model, the sensors' ranges and locations within the robot are inferred. The features map stored on the RoboEarth environment allows the visual SLAM to provide a continuous robot localization when the map is reused. Using all this information, a list of robot locations is computed from where the object is likely to be detected.

Upon execution of the generated CRAM plan, the robot sequentially navigates towards the computed locations from where it searches for the object until it is found. The detected object is added to the initial semantic map, which is then finally uploaded back to the RoboEarth database.

4.5. Robot Capabilities for Active Perception

Since RoboEarth aims at knowledge exchange among heterogeneous robots, we cannot assume that every robot possesses all required capabilities for executing a recipe. Therefore, both those capabilities that are available on a robot and those that are needed for a task are modeled and can automatically be matched using the SRDL. This procedure is described in detail in Tenorth et al. (2013). Capabilities are usually *provided* by software components (e.g. ROS nodes) on the robot, are *interfaced* from CRAM plan fragments, and are *described* in SRDL to allow reasoning about which tasks are feasible. Capabilities are often not binary, but may be available to a certain degree. It is however hard to measure this, since the criteria will be different for many different abilities. We therefore do not store a quantitative degree to which an ability is available, but distinguish different cases as specialized subclasses as can be seen e.g. for the different kinds of navigational abilities. In general, SRDL does support numerical attributes such as the range of a laser scanner or the resolution of a camera. Dependencies of actions on capabilities are usually not described in the recipe itself, but inherited from more generic action classes in the RoboEarth ontology (e.g. that all kinds of reaching motions need an arm component). Capabilities are also described as OWL classes and are declared in another branch in the RoboEarth ontology. The capabilities needed for the two recipes described in this work that focus on active perception are highlighted in figure 4.4 and will be presented in the remainder of this section.

CollisionFreeNavigationCapability represents the ability to safely navigate to a goal. The initial global navigation plan to achieve the goal is locally modified by a reactive navigation module which is responsible for computing the motions finally commanded to the robot. The planning technique is based on a A*-type algorithm Latombe (1991). For reactive navigation, we have applied ORM

Algorithm 1 SemanticMapping(**in:** environType, environId)

```
subDataBase = load-typical-object-models(environType)
slamVisualMap = void
start-exploration-modules()
start-vision-modules(slamVisualMap, subDataBase)
repeat
  explore-environment(freeFrontiers)
until check-finished(freeFrontiers)
return-to-initial-pose()
environment = upgrade-to-semantic(slamVisualMap)
upload-environment-map(environment, environId)
```

Algorithm 2 ObjectSearch(**in:** environId, object)

```
environment = download-environment(environId)
semanticMap,slamVisualMap = extract(environment)
start-navigation()
start-vision-modules(slamVisualMap)
nextPoses = infer-likely-locations(semanticMap, object)
repeat
  go-to-next-best-pose(nextPoses)
  slamVisualMap = search(slamVisualMap,object)
until check-if-object-found(object) or last-location-reached
stop-vision-modules()
environment = upgrade-to-semantic(slamVisualMap)
upload-environment-map(environment,environId)
```

Minguez (2005) adapted for differential drive robots due to its performance in dense, complex and cluttered environments. A Rao-Blackwellized particle filter Grisetti et al. (2007) is used to estimate the robot location and the 2D navigation map from 2D laser rangefinder readings.

EnvironmentExplorationCapability declares the ability to actively build a 2D navigation map of an unknown environment. Based on the 2D laser readings and the odometry, the component guides the robot in building a 2D map of its environment. The main issue is to compute at run-time the next robot locations from where to perceive unexplored regions. The next point of view is computed according to the frontier-based approach Yamauchi (1997), where the robot moves while avoiding obstacles and integrating the 2D laser readings into the map (*NavigationComponent*). The exploration ends when the map contains no more accessible frontiers. Figure 4.5 visualizes the method.

ObjectRecognitionCapability declares the ability to recognize objects in single images and to provide an initial estimate of their 3D location with respect to the camera. The corresponding component implements an object recognition algorithm Civera et al. (2011) in which each object is modeled as a collection of *faces*. Each face comprises an image that represents a point of view of the object, a set of SURF features Bay et al. (2008) and their associated 3D coordinates in the local object frame, obtained by multi-view geometry Snavely et al. (2006). These models are initially stored in the RoboEarth database. When a subset of them is required to fulfill a task, they are downloaded, creating a local subdatabase used by the recognition algorithm.

VisualSLAMCapability declares the capability to estimate a visual SLAM map composed of point features and recognized objects, and a 3D occupancy grid map. This capability is implemented by a distributed framework, C^2TAM Riazuelo et al. (2014b) proposed in chapter 3. A lightweight process handles the camera tracking on the onboard robot computer, while the expensive map optimization is externalized as a service in the cloud (Amazon EC2 service Amazon Inc. (2012)) using the RoboEarth Cloud Engine Hunziker et al. (2013). More details about the use of Amazon EC2 can be found in apendix B. Thanks to this division, the hard real-time constraints mandatory in a robotic embedded system are met by the visual SLAM, despite the typical network delays in the link with the cloud server. The SLAM map not only includes visual point features but also objects that are recognized in the images by the *ObjectRecognitionComponent*. The recognition models come from the local

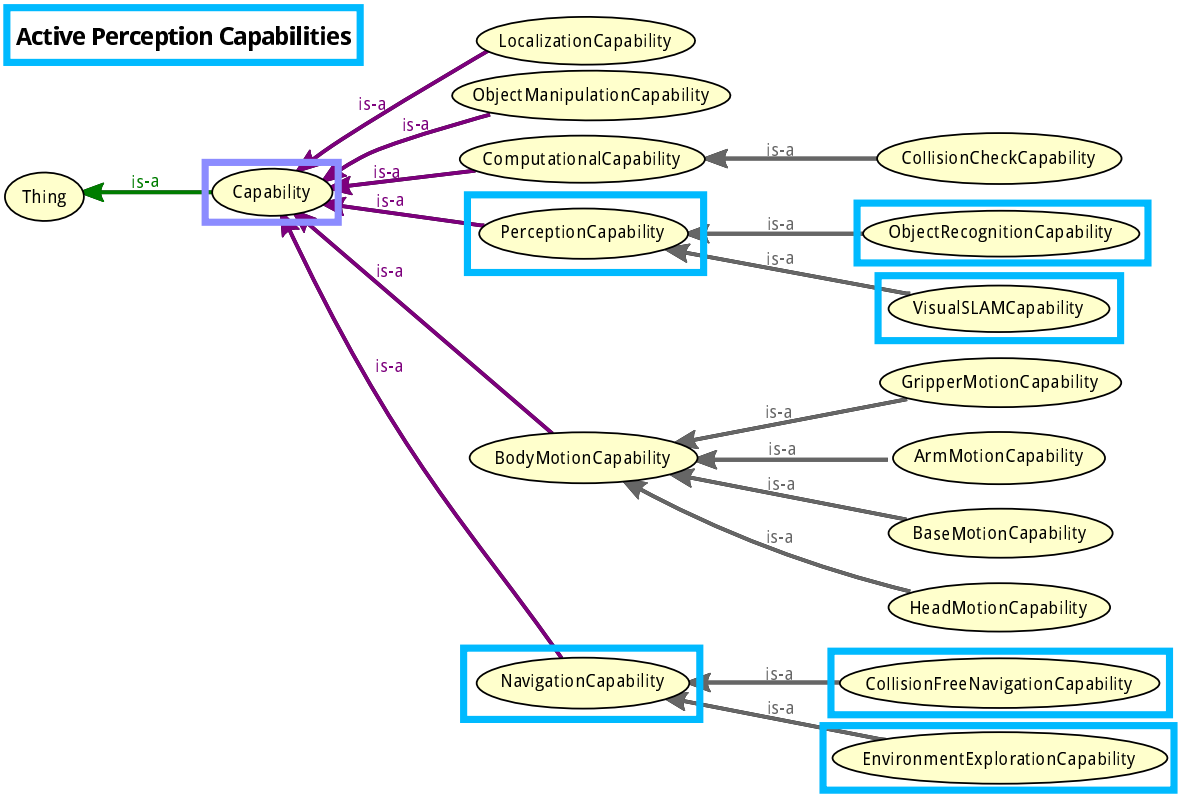


Figure 4.4.: A sub-branch of the SRDL ontology stating some of the robot capabilities. All the mandatory compatibilities for active perception are highlighted in blue.

subdatabase provided by RoboEarth. Once the map is computed, the result is incorporated into the RoboEarth environment data structure, providing the data for the following classes:

- *SemanticEnvironmentMaps* are described in OWL and consist of objects detected in the environment, described as instances of the respective object classes in the ontology. This allows the application of logical inference methods to the spatial configuration of objects. The object instances may further contain information about their extensions, 6D poses and possibly CAD models describing their geometry and appearance.

- *OctoMap*: a 3D occupancy grid map, coded as proposed in Hornung et al. (2013). It is computed from RGB-D sensor readings in the visual SLAM keyframes. This map can be reused to generate 2D maps for navigation.

- *ReVslamMap*: the raw storage of visual maps for visual localization. They are built from the sole input of an RGB-D camera. Provides a continuous localization of the robot while the map is building. Furthermore, thanks to the capability of Roboearth system for sharing environments, this map can be downloaded and reused by other robots in order to localize, while navigate, in the same environment.

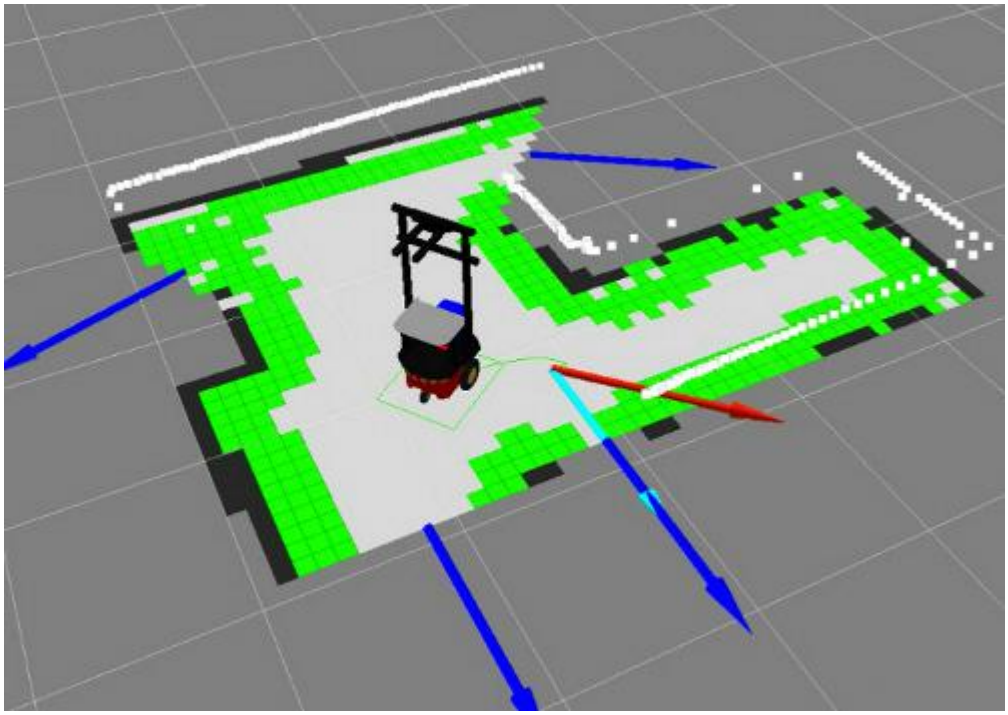


Figure 4.5.: Visualization of the frontier-based exploration algorithm. Black contours represent known obstacles and the green grid cells encode the inflation for safe navigation. The dark blue arrows represent unexplored frontiers. The next frontier to be explored is coded as a light blue arrow.

4.6. Reasoning about Object Locations

In order to successfully find an object in the environment, a robot must answer the questions "Where is the object likely to be?" and "Where do I need to go in order to see it?".

Inferring likely object positions

We employ knowledge that has been extracted from the OMICS common-sense database Gupta and Kochenderfer (2004) and converted into the representation used in the robot's knowledge base Kunze et al. (2010) to compute likely object positions. The OMICS database holds tuples of objects and their locations in the form $(object, location)$. The number of times a relation is contained in OMICS can be used to approximate the likelihood that an object O can be found at a location *LOC*:

$$P(O|LOC) = count(O, LOC)/count(LOC) \tag{4.1}$$

where *count* is the number of database entries. The value of $P(LOC|O)$ is calculated from the above model using Bayes' rule. To retrieve the location with the highest probability we simply apply the *argmax* operator

$$\underset{LOC \in Locations}{\operatorname{argmax}} \quad P(LOC|O) \tag{4.2}$$

The resulting models allow queries for the locations of objects given by corresponding landmark objects. These object classes can be grounded in the robot's semantic environment map to determine their positions.

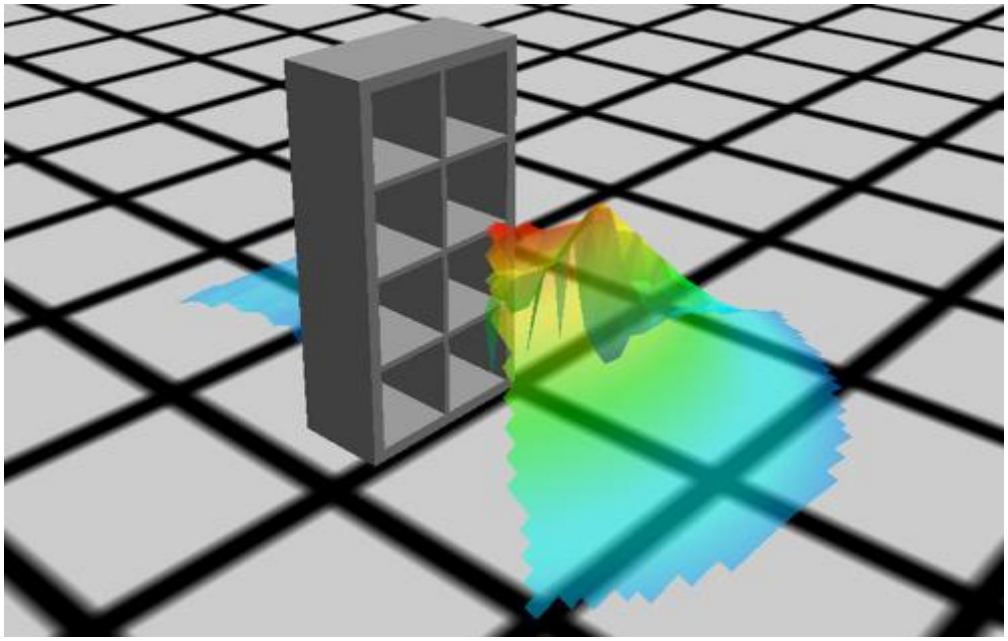


Figure 4.6.: Visibility costmap computed from the semantic environment map, the semantic robot model and geometric object models downloaded from RoboEarth. The colors indicate the amount of the object that is visible from a given camera of the robot considering its pose.

Computing robot poses using visibility reasoning

Based on the semantic map (that contains known object instances in the environment) and CAD models of these objects previously downloaded from RoboEarth, the system computes a visibility costmap describing from which robot poses the object is likely to be visible Mösenlechner and Beetz (2011). Especially for objects that are inside a cabinet or shelf, occlusions by the surrounding objects need to be taken into account when planning a pose for the robot. To compute the costmap, the system renders the scene from the viewpoint of the inferred object location and computes the amount of the object that is visible from each grid cell in the costmap (fig. 4.6).

4.7. Experiments

This section is devoted to showing how diverse robots benefit from the cloud-based RoboEarth semantic mapping system. The experiments include those carried out with a real Pioneer P3-DX robot and simulations. We demonstrate how a simple robot can reliably and efficiently build and exploit the semantic maps needed to perform quotidian tasks using the Roboearth cloud services [*]. The experiments are based on the two action recipes described in section 4.4.

Assuming that RoboEarth contains a huge database of object models, one key advantage is the ability to serve a reduced subdatabase that only contains the relevant models for the current tasks. This reduces the local computation overhead and improves recognition precision and recall. In the case of the semantic map building for a novel environment, given the type of the environment, in this case

[*] https://youtu.be/ehVt-eqk3dI

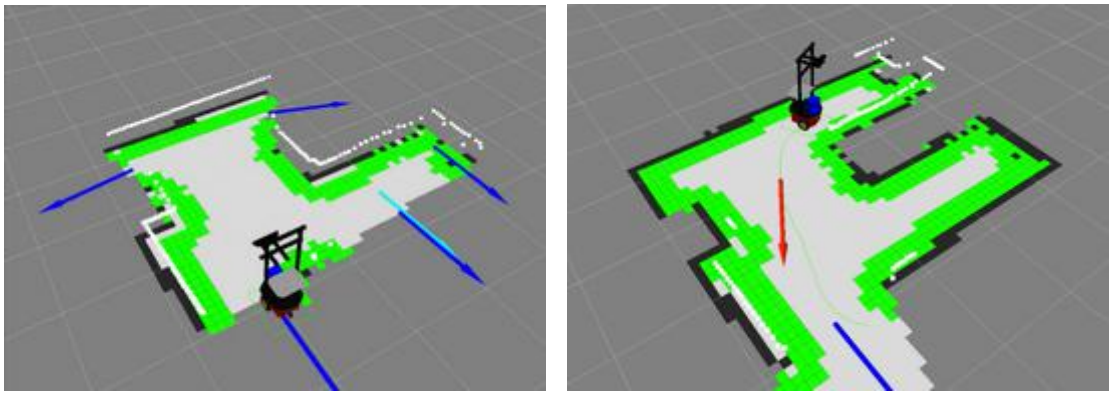


Figure 4.7.: Initial (left) and final (right) steps of the exploration algorithm. The dark blue arrows represent the currently unexplored frontiers.

a hospital room, RoboEarth is able to elaborate and serve to the robot a subdatabase containing only object models expected to be relevant and salient in this environment. In this case, the selected object categories are a bed and cabinet. For each object category all the relevant individual object models are included in the subdatabase. In contrast in the active search recipe, the served subdabase would contain only the recognition model of the object searched for.

Given the SRDL model of a robot, RoboEarth can produce and serve a customized CRAM plan for this robot. In the simulation we consider two different robots operating in two different environments. It is shown how, from a single recipe, four different execution CRAM plans are generated, one per robot-environment combination.

To illustrate the benefits that a simple robot can gain from using RoboEarth, we focus on the increase in efficiency derived from the exploitation of the knowledge-based reasoning available in the semantic maps in the case of a search for a novel object. In contrast to an exhaustive search, RoboEarth exploits a map the environment acquired previously and performs a knowledge-based search strategy of small objects by landmark objects.

4.7.1. Real-world experiments

The following scenario has been investigated: A robot in a hospital room has to find a bottle to be served to a patient. Initially, the robot does not know the location of the bottle. The naïve and expensive solution would have been to exhaustively search the whole room. In contrast, to improve efficiency, we use both the semantic mapping and object search recipes (Alg.1, Alg.2) to embody a knowledge-based search strategy in the robot.

We used a Pioneer P3-DX in which the navigation is based on a Sick 2D laser scanner and odometry sensors. It has been implemented by means of the ROS stacks *GMapping*, and *move_base* that has been extended to include the ORM obstacle avoidance. The robot also incorporates a Kinect RGB-D camera that provides the raw data for visual mapping. The visual SLAM is implemented by means of the C^2TAM algorithm that externalizes heavy computations using a Platform as a Service, in our case the RoboEarth Cloud Engine. It is worth noting that during the experiments, C^2TAM has been able to fulfill all the mandatory real-time constraints of our robot-embedded computer, despite the delays and low bandwidth typical of any computer network. Regarding the inference methods, the ROS RoboEarth stack ROS RoboEarth (2014) and the KnowRob knowledge base Tenorth and Beetz

Figure 4.8.: Object recognition events: bed (left) and cabinet (right).

(2013) are used. The capability matching and the CRAM plan generation have been executed locally on our robot computer, but these can also be externalized to the cloud.

Semantic Mapping

Before performing the task, the knowledge base infers that the bed and the cabinet are likely landmark objects, and the corresponding object models are inserted in the model subdatabase that is served to the robot. A customized CRAM plan is generated for the robot based on the recipe. The robot executes the CRAM plan and starts to explore the unknown environment until it obtains a complete map of the room. Figure 4.7 shows the beginning and end of the exploration. At the beginning the map is incomplete, with several open frontiers that have yet to be explored. At the end the complete map is estimated.

While the robot is exploring the environment, the perception component builds the visual SLAM map and inserts the detected objects according to the models in the subdatabase. Figure 4.8 shows two examples of object recognition events. Once the exploration is finished the robot uploads the created semantic map to RoboEarth (as we can see in fig. 4.1). This comprises the detected objects (fig. 4.9), the map of visual features,and a 3D occupancy grid map, coded as an OctoMap (fig. 4.10).

Object Search

The second recipe execution presents the guided object search. This is based on the semantic map of the environment built and uploaded in the previous exploration (Sect. 4.7.1). The object location inference determines the cabinet as the landmark object to guide the search for the bottle. Taking the scene layout and occupancy map into account, several reachable robot locations from where the bottle is likely be detected are computed (see figure 4.11). Considering the SRDL description of the Pioneer P3-DX, the stored semantic map and the action recipe, RoboEarth provides: 1) a custom 2D map for navigation estimated from the OctoMap, 2) a customized CRAM plan, and 3) the set of recognition models for the bottle.

The provided CRAM plan iteratively drives the robot to a list of selected positions until the object is eventually found. Once it is located, its position is added to the map and the map is uploaded to RoboEarth. Figure 4.12 shows the robot trajectory and the semantic map including the objects known a priori (bed and cabinet) and the new one (the bottle).

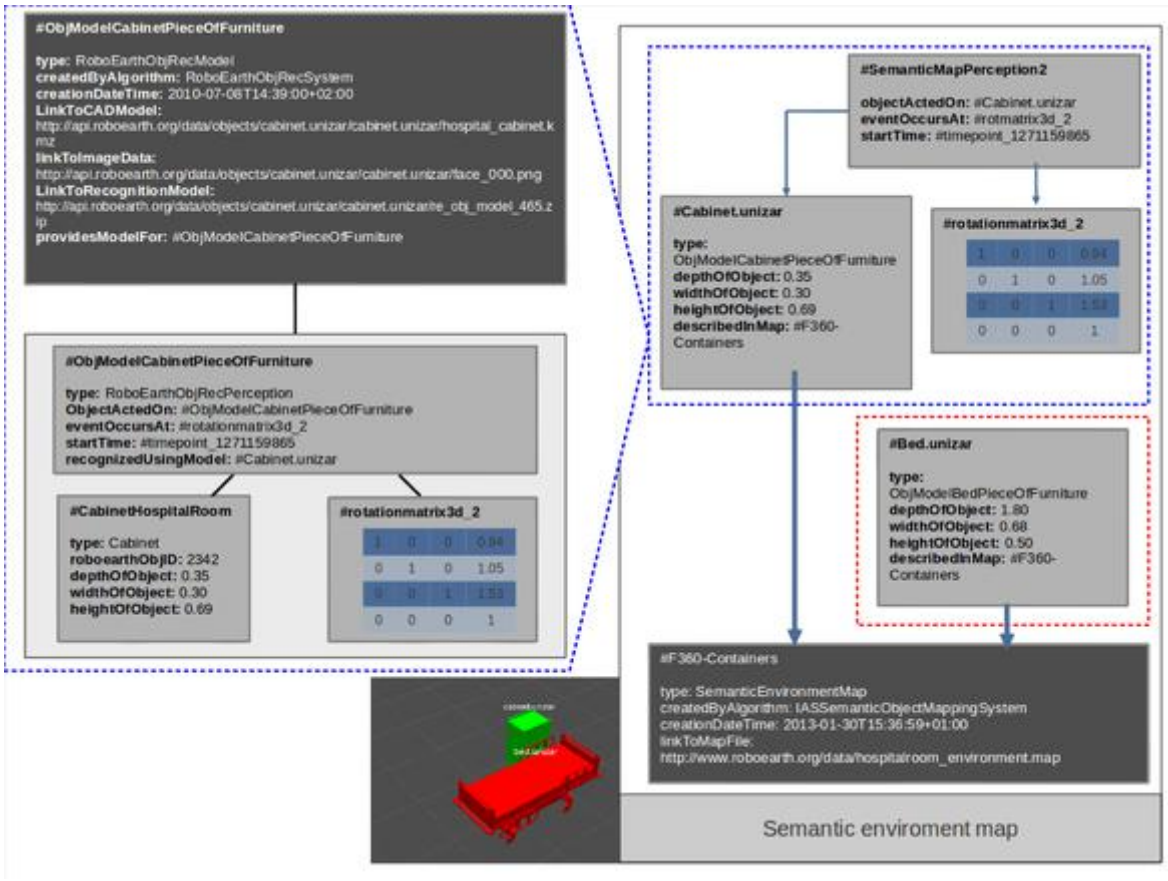


Figure 4.9.: Detailed storage format for a semantic map composed by two objects, a bed and a cabinet. Each object instance contains information about the type of object, dimensions, recognition model used, time detection and its location into the map.

4.7.2. Simulation Experiments

The goal of the simulation experiments is to demonstrate the interoperability of the system. For these experiments, we have used the open source robotics simulator Gazebo Koenig and Howard (2004). The same object search action recipe has been executed on two different robots in two different environments. The selected robots have been the holonomic service robot Amigo Lunenburg et al. (2012), and the previously described non-holonomic Pioneer P3-DX. Per each selected robot, the SRDL model describes its capabilities and kinematics, enabling RoboEarth to produce a specific CRAM plan for a definite robot in a particular environment.

The searched object has been considered to be probably found on top of beds, cabinets or shelves. The first environment emulates the hospital room, assuming a semantic map where a bed, and a cabinet have been located. The second environment mimics a suite, composed of two rooms communicated through an open door. It is assumed to have a semantic map where a bed, a cabinet and two shelves have been detected. Figure 4.13 shows the paths resulting from the four different CRAM plans, one per each robot in each room. The paths mirror the difference in locomotion, the Amigo robot is able to maneuver more efficiently in the tight spaces than the non-holonomic Pioneer P3-DX. The two robots also have their camera in different locations; consequently the reasoning for the path generation has

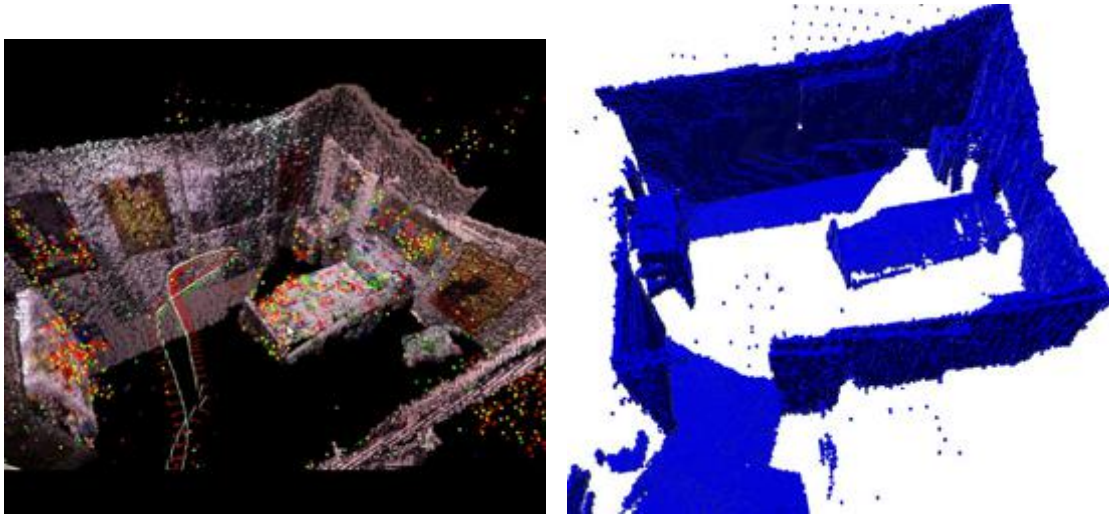


Figure 4.10.: Map of visual features (left), and 3D occupancy grid OctoMap (right).

been influenced by the differences in visibility.

4.7.3. Performance Improvements

The purpose of this experiments is to highlight the benefits of using the proposed system. We focus the quantitative results on three aspects: (1) the externalization in the cloud of the most intensive computations; (2) the use of a subdatabase of objects which improves recognition; (3) the efficiency of the knowledge-based search strategy based on landmark objects implemented by the object search action recipe.

Computational Efficiency

The use of the cloud for externalizing the expensive computation processes provides an improvement in the response time, as we can see on figure 4.14. This figure presents two graphs that show the response time per frame of the tracking process of the visual SLAM system with respect to the size of the map during the execution of exploration action recipe. On the top graph, we can see the performance of the system when the expensive computation process of the visual SLAM system used is running on the cloud. The tracking response time remains constant (around 10 ms) independently of the map size. The bottom graph shows the tracking time when the complete C^2TAM system is running onboard the robot. We can see how the tracking response time increase when the size of the map grows and even it overtakes the video frame rate threshold (33 ms). We can conclude that the externalization of the expensive map optimization process of C^2TAM as a service in the cloud provides an improvement in the response time of the real-time critical processes because they can benefit of all the onboard resources once the mapping process is outsourced to the cloud.

Recognition using a Subdatabse

Two search strategies has been tested for a range of subdatabase sizes. The first strategy is a naïve detection that checks all the models in the subdatabase. The second is an advanced one that only

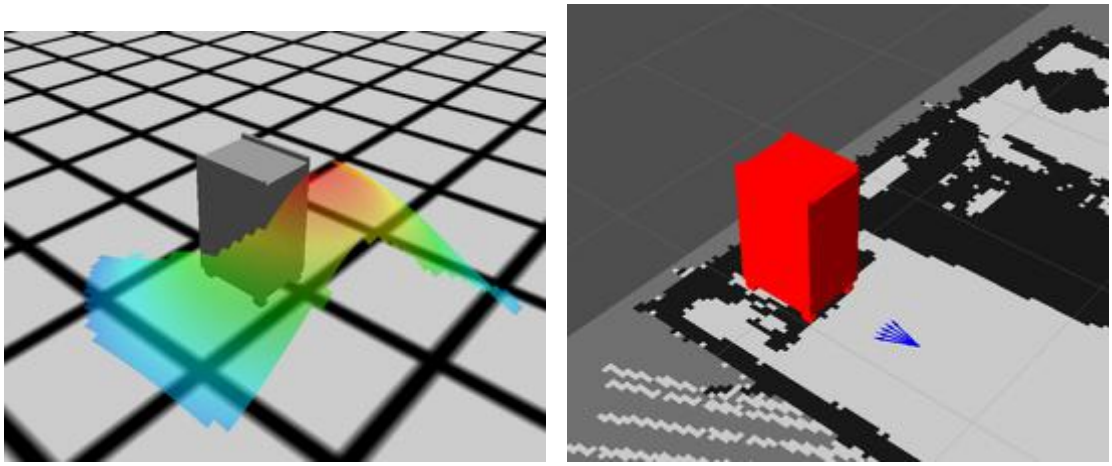


Figure 4.11.: Visibility costmap (left). Occupancy map and, in blue, the selected search robot locations (right)

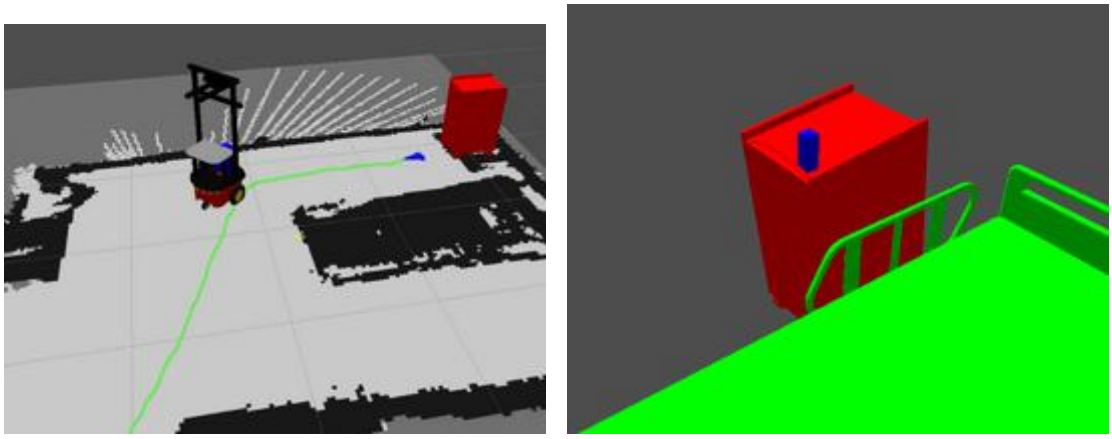


Figure 4.12.: Computed robot locations for detecting the object, in blue, and planned trajectory, in green (left). Final semantic map including the bottle detected on top of the cabinet (right).

checks the 10 most promising object models according to an appearance score obtained when their local features are converted into bags of words (BoW) Gálvez-López and Tardós (2012). In this experiment we focus on the semantic mapping of a hospital room. The subdatabase contains the RoboEarth provided relevant models in all the experiments. Additional object models, up to 500, are added to the subdatabase to analyze the effect of a big database containing objects not appearing in the actual scene. Figure 4.15 shows a quantitative performance analysis. The top graph shows the naïve detector, bottom graph shows the advance BoW recognition. As expected mean time of detection after the BoWs preselection scales better with the subdatabase size. Both of the methods produces more detections with reduced subdatabase, i.e. low false negative rate after processing the whole experiment sequence. Additionally in our experiments we did not detect any false positive, what is a good indicator of a remarkable recognition precision. In any case, in both algorithms, we can see how the increase number of objects in the subdatabase degrades the performance. We can conclude

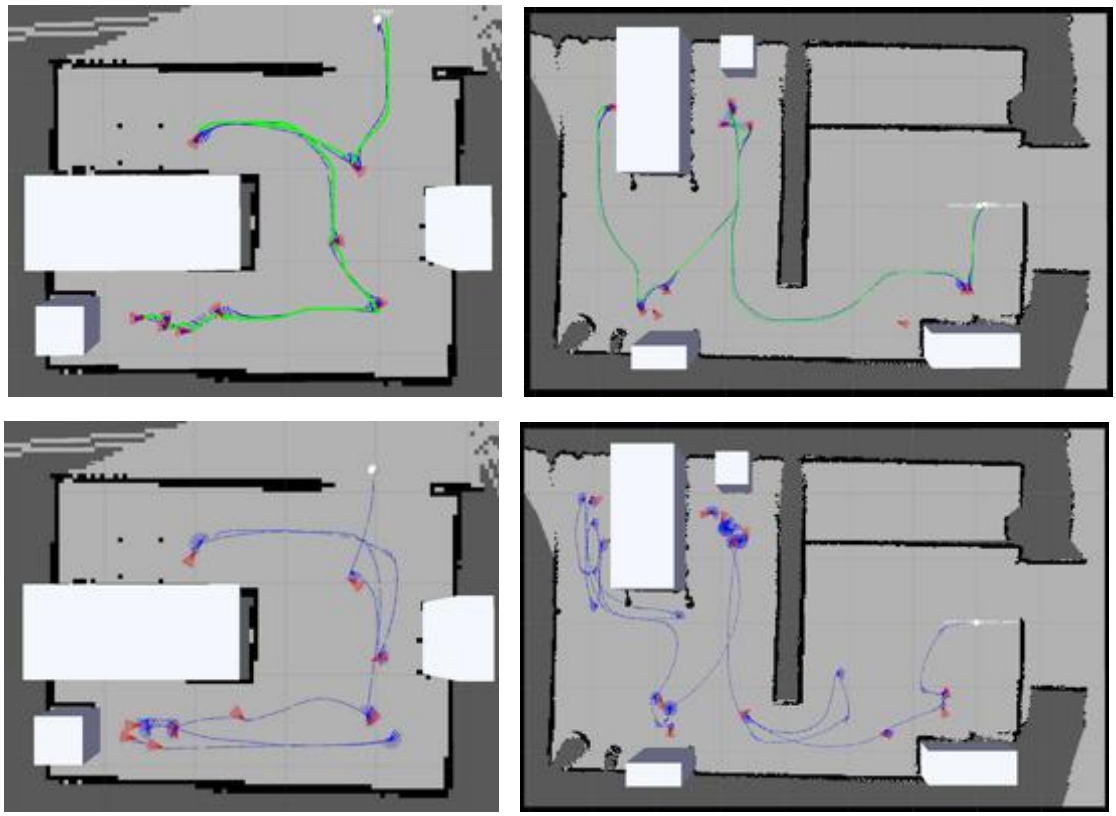


Figure 4.13.: Travelled path (blue) in simulated object search. Top row displays Amigo, and bottom row displays Pioneer. Left column for the room, right column for the suite.

that a subdatabase which only contains the most relevant models for a specific task provides a better performance on the object detection in terms of number of detections and speed. RoboEarth is able to provide this subdabase of only relevant objects.

Knowledge-based Search Strategy

Finally, in terms of the proposed knowledge-based search strategy, we show how the priors provided by the semantic map are able to reduce the number of potential search locations, and hence significantly reduce the search time. We compare the guided exploration trajectories with those of an exhaustive search. The comparison is made in terms of the number of locations from where the object search is performed in the worst case. The selected scenarios are the room and the suite described in the previous section. For the search locations in the exhaustive case, we have selected the Art Gallery algorithm Shermer (1992) because, for a given sensor visibility range, it provides the minimum number of positions which can cover a particular environment. Figure 4.16 shows the locations which achieve full coverage of the considered environments. The number of locations depends on the size of the environment – the bigger the environment, the higher the number. In our case, 40 locations were computed for the room and 100 for the suite. The benefit is evident if we compare this with the knowledge-based search (fig. 4.13 bottom row), where the room needs only 9 locations and 15 were needed by the suite, leading to a corresponding reduction in the search time.

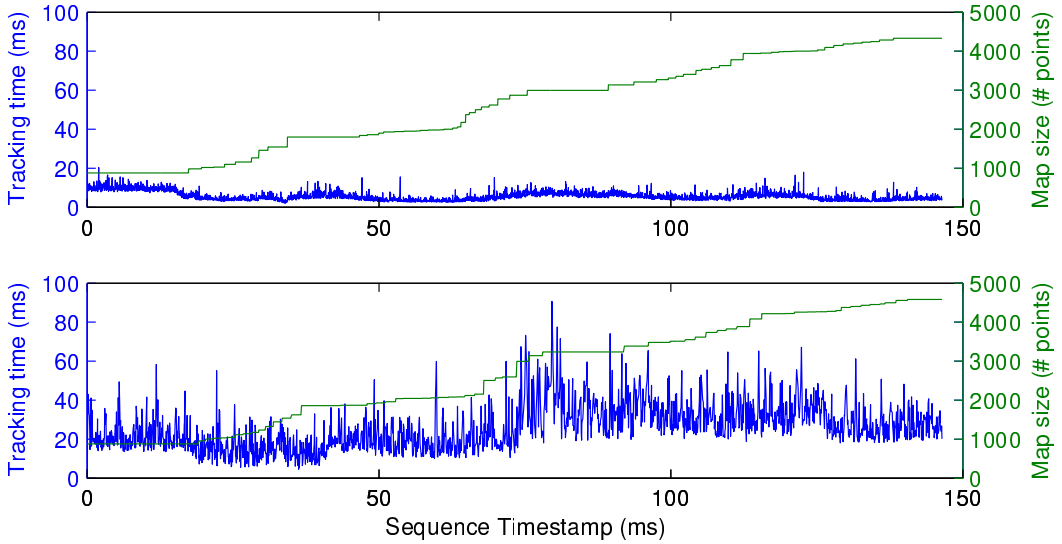


Figure 4.14.: Response time of the tracking process. Top graph C^{2}TAM running mapping in the cloud, botton graph all C^{2}TAM processes running onboard the robot.

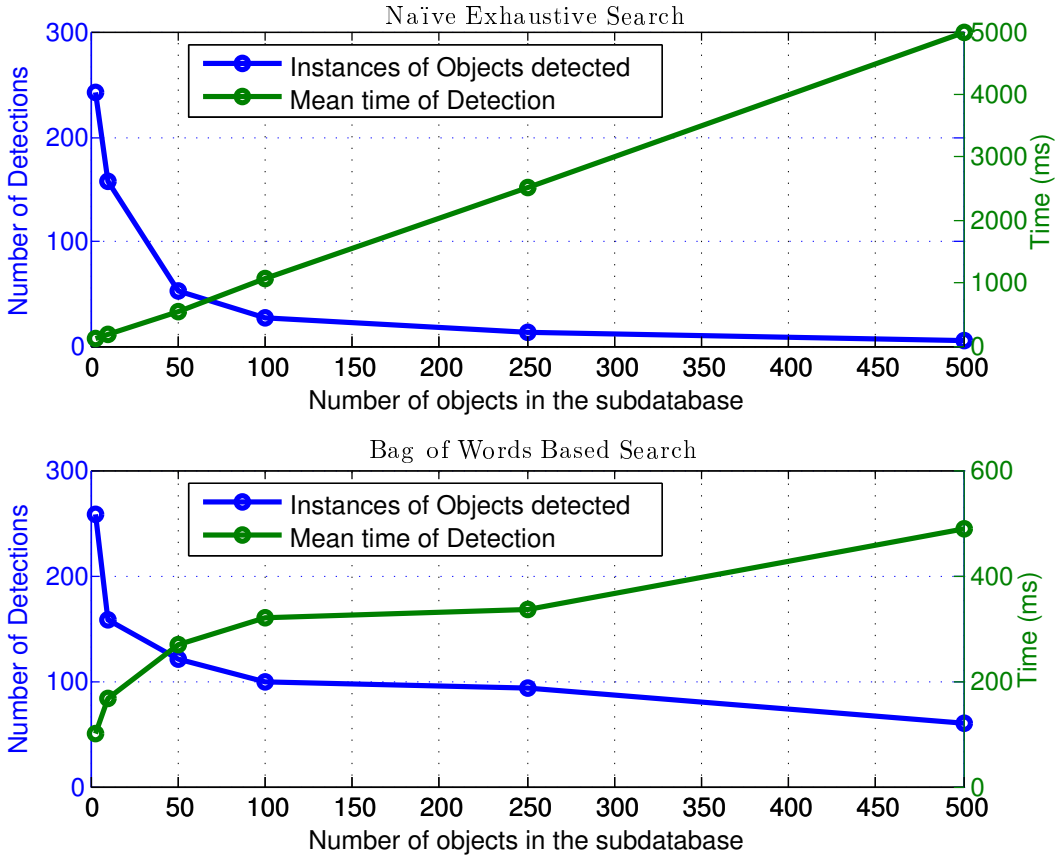


Figure 4.15.: Number of detections and time performance of the object detector as function of the subdatabse size. Top, naïve recognition. Bottom Bag of Words preselection

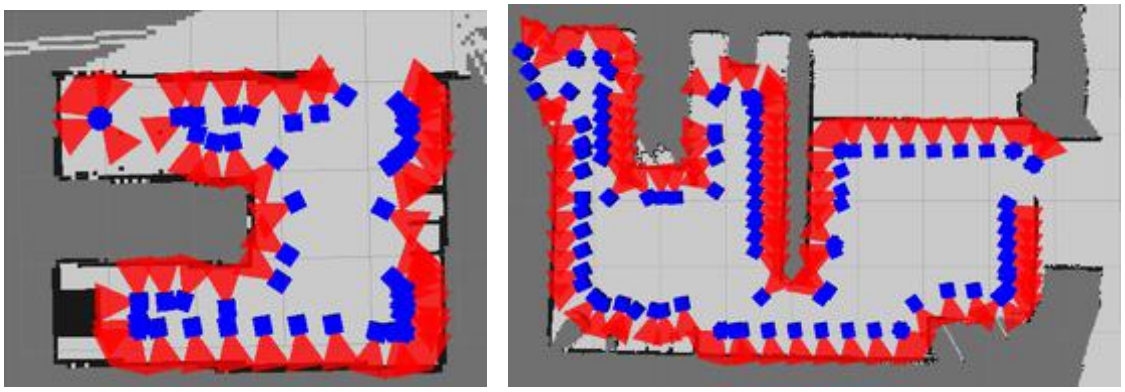


Figure 4.16.: Art Gallery exhaustive search. Blue squares code the search locations, and red sectors represent the camera field of view. Room: 40 locations (left). Suite: 100 locations (right)

4.8. Discussion

A robot operating in an environment for the first time can benefit from information previously stored by other robots operating in the same environment, thanks to the RoboEarth semantic mapping system. The proposed semantic mapping system combines a visual SLAM map of objects with an ontology representing the knowledge. Thanks to this combination, knowledge-based reasoning about map entities becomes possible.

We have demonstrated that the building and exploitation of this mapping system can be implemented as web and cloud services. The robot has to provide its SRDL description, and hence RoboEarth provides all the information needed to execute the task. The result of the execution is also stored in the database for reuse by the same or other robots. We have provided a pioneering experimental validation of a web-enabled cloud semantic mapping system exemplified in the case of map building and guided search for a novel object. We conclude that our system can (a) enable robots to perform novel tasks, (b) generate semantically meaningful environment maps, and (c) reason about these maps in conjunction with formally described background knowledge. Indeed, the strategy cannot address all cases in an open world at once (i.e. recognize all objects at all times), but this is in general not feasible at the moment. With limited on-board resources, the options are either to manually select a number of objects that can be recognized (as it is commonly done today), or to give the robot the ability to autonomously select a range of object models that are to be expected in the environment. This is obviously limited by the quality of the predictions, but still more flexible than a rigid selection of objects.

Evidence has also been provided about the possibility of externalizing in the cloud those processes which are demanding in terms of memory or CPU while at the same time meeting the hard real time robot constraints. We can hence conclude that the operation of simple robots with typical computing and networking facilities can be boosted by RoboEarth.

Chapter 5

Semantic Visual SLAM in Populated Environments

5.1. Introduction

In contrast to the environments used to validate semantic navigation in the previous chapter, the service robots usually works in populated environments. The contribution of the work presented in this chapter is to design and implement a visual SLAM system able to perform robustly in populated environments. The system has been evaluated using two visual SLAM algorithms based on keyframes: C^2TAM presented in chapter 3 and ORBSLAM2 Mur-Artal and Tardos (2016).

State of the art visual SLAM algorithms rely heavily on the rigidity prior, which assumes that each image is filled with a that is mostly rigid and persistent. To exploit the rigidity prior, the algorithms include a RANSAC-like voting stage to blindly detect and remove any dynamic element from the mapping process. This delivers a nice robust performance in mostly rigid scenes. Unfortunately, the observation of populated environments implies non-persistent scene changes and scene regions with severely non-rigid motions. The resulting images might be filled with non-rigid and non-persistent elements where the voting algorithms will fail.

Our proposal is to detect and track people in each frame of a video stream (See fig. 5.1), Our first contribution is to mask out of the SLAM processing those image regions corresponding to human activity. Hence, we restore the validity of the rigidity prior for the non-masked image regions that can now be processed successfully with the standard VSLAM algorithms. The masked frames are fused in a map that only represents the geometry of the non-human rigid elements in the scene. We refer to this as the *unpopulated map*. In our proposal, the unpopulated map is a dense occupancy grid computed as an Octomap Hornung et al. (2013) resulting from the fusion of a selected set of RGB-D keyframes which are accurately located in a common reference after the Bundle Adjustment of a sparse point feature matches Riazuelo et al. (2014b); Mur-Artal et al. (2015).

Our second contribution is to exploit the human detection and tracking in each frame of the video stream to build a *semantic human activity layer* that registers the trajectory of each person entering within the limits of the unpopulated map. This is the result of combining the accurate camera location that the unpopulated map provides with the human detection in real-time at frame rate. The scene is observed with a mobile RGB-D camera, but the people trajectories are estimated on the map, irrespective of where the camera was when the person was imaged (see fig. 5.1.) The semantic layer can be exploited for autonomous robot navigation and planning in populated environments.

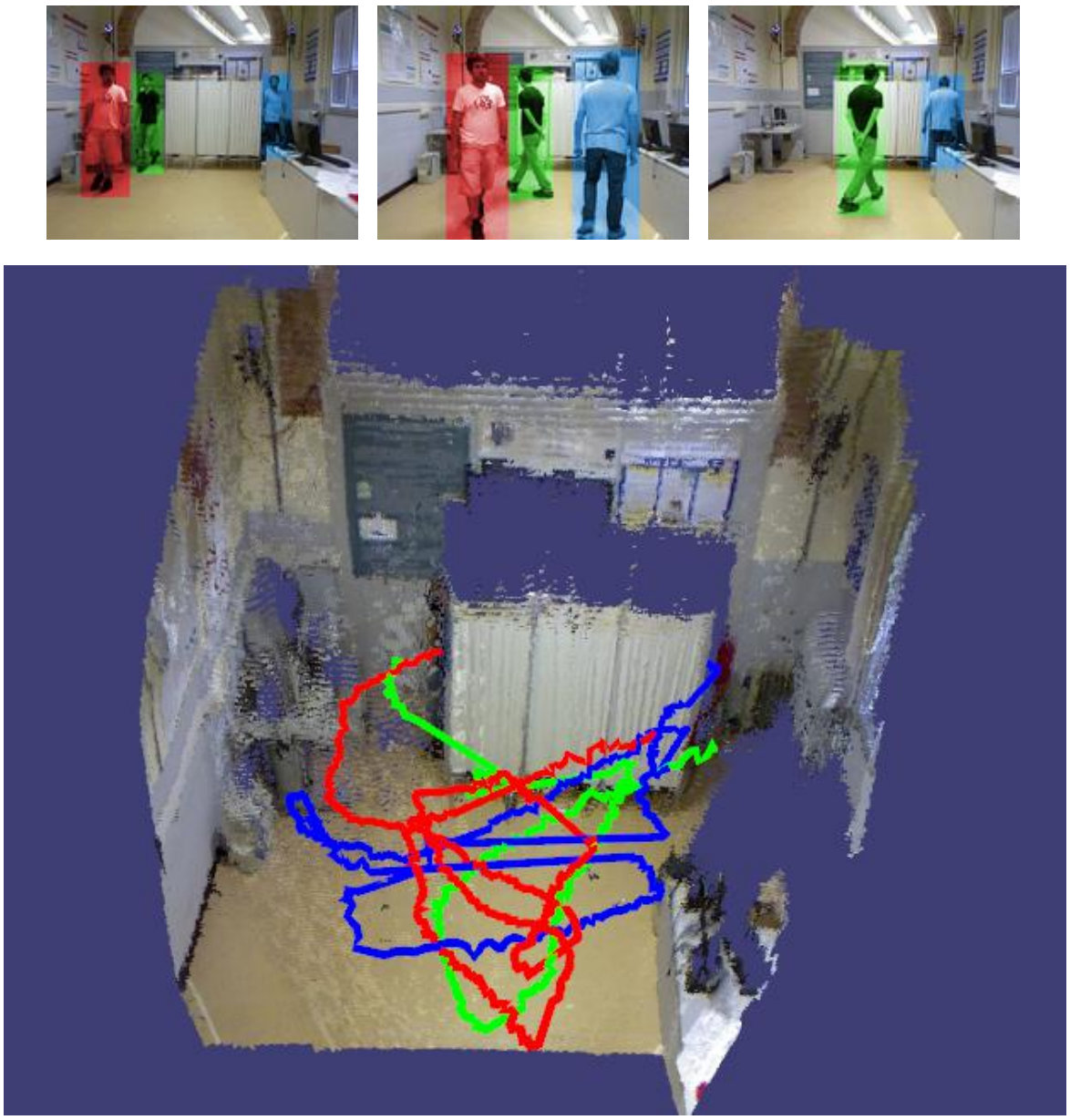


Figure 5.1.: (Top row) Typical frames of a populated scene. The human tracker detections are overlaid in colours identifiying the person. (Bottom row) the two layers map. The geometrical unpopulated map is a dense occupancy map after removing people populating the scene. The human activity semantic layer is the set of trajectories of the detected persons with respect to the unpopulated map.

The rest of the chapter is organised as follows: Section 5.2 discusses related work. Section 5.3 describes our system. In section 5.4 we present the experimental results and section 5.5 concludes and presents lines for future work.

5.2. Related work

Most semantic mapping approaches are focused on static objects instead of moving objects. In Nüchter and Hertzberg (2008) and Goerke and Braun (2009) a 3D laser scan is used for enhancing the map with semantic information. Using 3D points from a stereo camera, Günther et al. (2013) builds a map and enriches it by inserting CAD models corresponding to the detected objects. Combining object recognition and visual SLAM in order to produce semantic maps has been extensively studied in recent years. Castle et al. (2010) and Civera et al. (2011) merged a monocular SLAM system and object recognition for enriching the map. In Gálvez-López et al. (2016) a combined approach is also presented, the authors not only compute the position of the objects in the map, but also add the objects to the optimization process. A recent work Salas-Moreno et al. (2013) incorporates the use of depth information for mapping and object recognition of object instances. Hermans et al. (2014) also uses RGB-D sequences, but goes further because the goal of the recognition is not object classes but categories. They create consistent 3D semantic reconstruction of indoor scenes and categorize each voxel in real-time, assigning a semantic label to the voxel according to a specified category. However, removing dynamic objects and annotating them is not a problem which has been studied in depth. OctoMap Hornung et al. (2013) blindly removes moving objects due to its probabilistic nature; it eventually filters out moving objects, but it does not contain semantic information about the removed objects. In contrast, our approach first identifies objects by categories and then removes them. It is even able to deal with stationary people. Newcombe et al. (2011) also deals with moving objects using a volumetric representation, but they do not add semantic information about these objects to the map. A preliminary work to identify and remove moving objects finding corresponding in two views is presented in Litomisky and Bhanu (2012).

Navigating in the presence of humans requires knowledge of people's movements. In Kruse et al. (2013) a collection of approaches to human-aware navigation are presented. Dondrup et al. (2015) presents real-time people perception framework, using detectors based on laser and RGB-D data and a tracking approach able to fuse multiple detectors.

The study of human motion patterns has become increasingly significant in recent decades. Wang et al. (2016) presents a method for classifying regions for human movements and Xiao et al. (2015) proposes a method to recognize and predict people's path using a pre-trained SVM as a classifier. In Wang et al. (2015) an approach for modeling the dynamics of human movements with a grid-based representation is presented. However on these methods the sensor remains always static in the scene in contrast to our approach.

Human detection and pose estimation in populated environments is a problem studied in considerable depth in the literature Viola et al. (2005), Leibe et al. (2005), Mikolajczyk et al. (2006). On the subject of articulation, Bergtholdt et al. (2010) proposes a fully connected graphical model for representing articulated models. Yang and Ramanan (2011) describes a general method for human pose estimation in static images based on a representation of part models. A generic approach based on the pictorial structures framework for articulated pose estimation is presented in Andriluka et al. (2009).

Wang et al. (2007) proposes a mathematical framework to integrate SLAM and moving objects whith the use of depth sensors. Ess et al. (2009) presents an approach close to ours as they mask detected people from the geometrical processing. However instead of building a full SLAM map they only compute visual odometry. Furthermore, additionally their focus is on people tracking while neglecting the scene unpopulated map.

Our proposal is a system that builds strongly on state of the art visual SLAM systems processing RGB-D images, C^2TAM Riazuelo et al. (2014b), and ORBSLAM2 Mur-Artal and Tardos (2016).

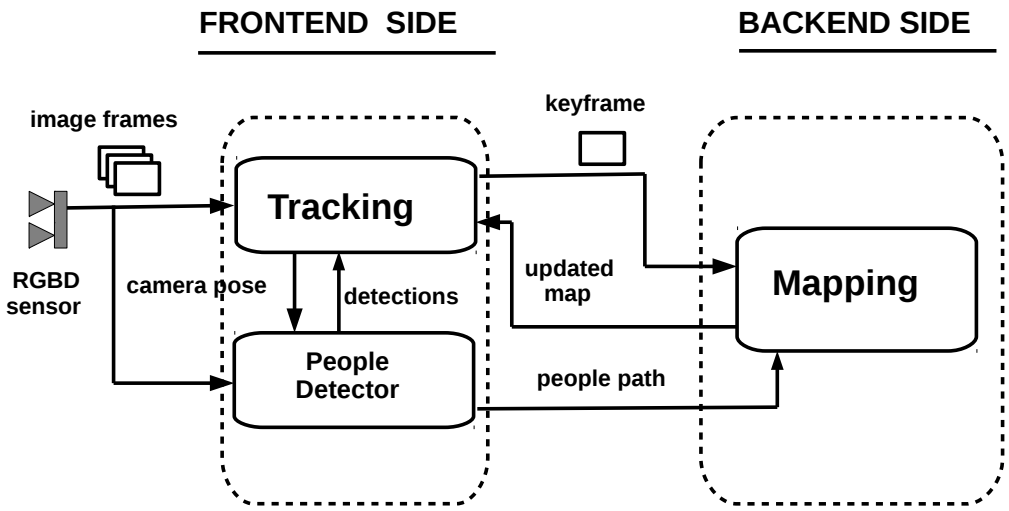


Figure 5.2.: Typical VSLAM parallel architecture and people detector integration.

Regarding people detection we integrate Jafari et al. (2014) because of its real-time RGB-D detection and tracking for mobile robots and head-worn cameras.

5.3. System Description

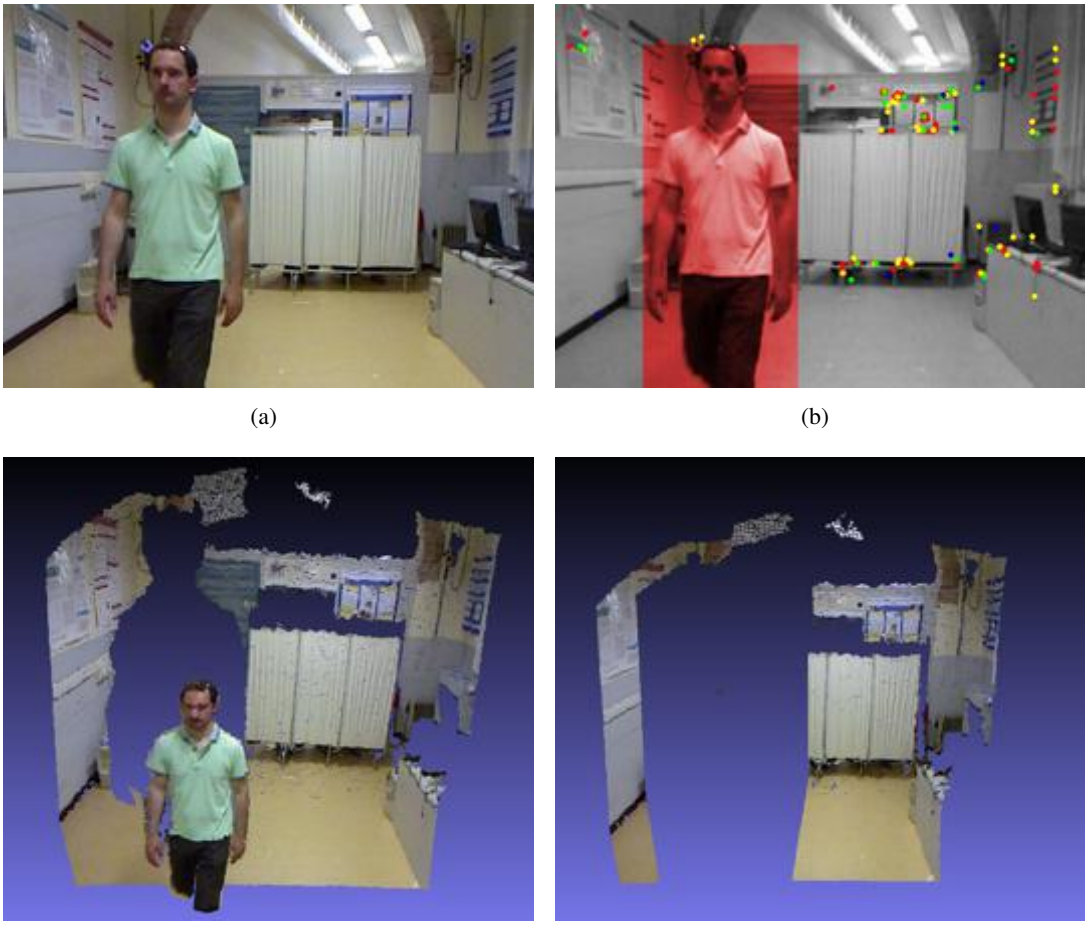
Klein and Murray proposed the ground-breaking PTAM architecture for purely RGB monocular sequences in Klein and Murray (2007). This is still the basis of state-of-the-art VSLAM systems also monocular Mur-Artal et al. (2015) or RGB-D C^{2}TAM Riazuelo et al. (2014b) or ORBSLAM2 Mur-Artal and Tardos (2016). Our proposal also builds on top the RGB-D ones. Although our approach is agnostic to the VSLAM approach, for the aim of simplicity we focus the system description on ORBSLAM2. The architecture is based on two intertwined processes running in parallel. The two processes are generically named as *frontend* and *backend*, see fig. 5.2. The frontend is focused on a minimal set of operations to provide the camera location in real-time, it assumes that the backend has provided a map of the scene. The backend concentrates the expensive mapping operations to estimate the scene map by means of non-linear optimization, it iterates at low frequency providing fast –but not real-time– updates of the estimated map.

The real-time camera pose is a tracking algorithm which assumes that the camera trajectory is smooth. If it gets lost, for example due to camera occlusion or motion blur, then the camera location has to be located from scratch before resuming tracking. This recovery stage is called *relocation* in the VSLAM literature, and it is also a standard component of contemporary SLAM systems.

Our proposal integrates a real-time people detection and tracking stage Jafari et al. (2014) in the system to enable its operation in populated environments. The people detection interacts with all the components and stages of the visual SLAM. We describe below the modified frontend, backend, relocation and human activity detection layer of the map.

5.3.1. Frontend process

The frontend processes each RGB-D camera frame in real-time to provide the camera location. Its main component is the *camera tracking*, which assumes that both an accurate sparse 3D point map of



(a) (b)

(c) (d)

Figure 5.3.: (a) Raw RGB. (b) Interest point detection and human activity masking. (c) Raw RGB-D depth channel. (d) RGB-D point depth channel after removing people depth points.

the scene and an estimated camera position, from which putative point matches between the current frame and the map are estimated. Then, the camera pose is estimated by non-linear minimization of the map reprojection error. The optimization is cheap because only the six d.o.f of the camera pose are optimized. The map points are not optimized because their estimate locations, provided by the backend, are assumed to be perfect.

The reprojection error includes a robust influence function Huber (1981) that marks as an outlier any match with a big reprojection error. If the outlier rate exceeds 50% of the computed matches, the camera location estimate is very likely to break down. Outliers correspond to mismatched points or to matches not following the rigidity prior. Hence, human activity makes the camera estimation more likely to fail because it generates outlier matches that increase the outlier fraction.

We propose to identify the frame regions which image human activity in order to mask them out of the camera tracking processing. The first step of the tracking is to detect interest points in the current frame, typically FAST Rosten and Drummond (2006). We additionally process the frame with a real-time people tracker Jafari et al. (2014) that yields a bounding box per each detected person. All the points of interest within the bounding boxes are masked out for the subsequent matching stage

(fig. 5.3 (b)). By masking out the FAST points on humans, we remove points that are probably outliers, reducing the proportion of outliers and hence increasing the probability of a successful camera location estimate.

The people tracker provides a 3D position of each detected person with respect to the RGB-D camera frame simply by reading the depth channel. Additionally, we have ready available each person trajectory referred to the static scene map frame, by just composing the person position with respect to the camera with the camera pose provided by the camera tracking. This mechanism to compute the people tracks referred to the map is the main ingredient of the semantic people activity layer of the map.

	Translation Sequence				Arc Sequence			
	ORBSLAM2		C^2TAM		ORBSLAM2		C^2TAM	
	ATE (mm)	% Frames Tracked	ATE (mm)	% Frames Tracked	ATE(mm)	% Frames Tracked	ATE(mm)	% Frames Tracked
No Masking	53.3	92.3	42.6	95.9	39.2	97.0	41.7	93.5
Ground-Truth Masking	15.9	90.7	16.7	92.3	19.6	98.6	28.6	94.5
People Detector Masking	22.6	92.5	20.2	94.3	21.2	99.2	29.1	87.5

Table 5.1.: System performance over KTP dataset sequences using ORBSLAM2 and C^2TAM.

5.3.2. Backend process

The backend concentrates all the compute-intensive or non-time critical operations derived from the mapping estimation. Its main component is a non-linear iterative optimization, after each optimization step, the map provided to the frontend is updated. It is only possible to achieve an update rate significantly lower than the frame rate, but fortunately it is acceptable.

The backend exploits the fact, well-known since the early times of photogrammetry, that given the points correspondences along a monocular image sequence both the camera poses and the 3D point positions can be estimated up to scale factor, for a rigid scene. The gold-standard algorithm is non-linear minimization of the reprojection error, known in the literature as Bundle Adjustment (BA) more specifically, the iterative optimization algorithm used is the Levenberg-Marquardt. We use an extension in the case of stereo cameras Mur-Artal and Tardos (2016).

PTAM-like algorithms exploit the fact that not all the frames in the sequence have to be included in the BA, but only a small fraction of them, known as keyframes. Different heuristics have been proposed for keyframe selection, for example in Klein and Murray (2007); Mur-Artal et al. (2015); Riazuelo et al. (2014b). The heuristics application results in a reduced set of keyframes that cover the imaged scene while having enough overlapping to ensure that all the points in the map are imaged from several images with a significantly different point of view in order to render parallax, ensuring a good geometrical conditioning for BA. There is overwhelming experimental evidence of the efficacy of these heuristics in the keyframe selection. The keyframe selection is made in the frontend.

The main challenges that BA has to face are the outliers, the local minima, and the high number of iterations to converge to the minimum. All these drawbacks can be overcome if an initial guess close to the solution is available. The intertwining between the frontend and the backend is responsible for mutually providing the necessary initial guesses for the non-linear optimization. When the camera explores new scene areas not yet included in the map, the frontend sends a new keyframe to the backend, with an initial guess for its location. A low spurious rate set of matches between the new keyframe and all the other keyframes already in the map is also available. From this initial information, matches for new points to expand the map can be robustly found exploiting the scene rigidity. Again, masking out the keyframe regions corresponding to human activity helps to reduce the outlier rate in the new matches, and hence increases the robustness. Given the new camera pose

estimate, initial guesses for the points newly added to the map can be also estimated. Thanks to the initial guesses the convergence for the newly added camera and map points is fast.

The sparse map M is the result of the BA. It is composed of a set of l keyframes $\{K_1, K_2, \ldots, K_l\}$ and n 3D sparse feature points $\{P_1, P_2, \ldots, P_n\}$. Per each keyframe, it estimates the camera position coded as the rigid transformation T_{WK_j} referred with respect to world frame W. The keyframe also contains a the camera depth channel of the image, which in contrast to the sparse points of the maps can provide dense depth information of the scene. Once we have accurate keyframe poses, T_{WK_j} from the BA, we use the Octomap algorithm Hornung et al. (2013) to fuse all the depth maps into a single occupancy map. Prior to the fusion, the depth regions corresponding to human activity are also masked out from the keyframes, so that the generated Octomap does not contain the people populating the scene. This people-removal process does not have to be in hard real-time, so it is performed in the backend and only on the keyframes. The unpopulated map could be easily reused in future reobservations of the same scene, irrespective of the present and future the human activity.

5.3.3. Camera Relocation

As mentioned above, relocation is mandatory after the loss of camera tracking, and it is necessary to be able to reuse a previously available map. The ORBSLAM2 relocation algorithm is currently the state of the art. Its first step is an indexed search for matches known as DBoW Gálvez-López and Tardós (2012). The search for matches between the current image and the available map mimics a query in a database, where the current image is the query, and the dabatase is holds the map points. In the second stage the camera pose is recovered using a RANSAC-like approach. In our proposal, during the relocation step, we mask out the detected people from the current image before querying the DBoW database. It has to be considered that in building the database, the regions corresponding to human activity were also masked out from the map, hence also from the database. We can conclude that human activity detection and removal extends the lifespan of map reuse, because the number of spurious matches is reduced both from the database and from the queries.

5.3.4. People detection and human activity layer

To deal with human activity we propose to integrate a people detection and tracking stage, in our case we use Jafari et al. (2014). We process all the images coming from the RGB-D camera. Each detected person in an image is coded as an image bounding box and a label is created that uniquely identifies the person. The 3D position of the person with respect to the camera is computed as the position of the center of the bounding box, and the depth channel. All these pieces of information are computed and stored in the frontend.

When the frontend decides to create a new keyframe, it sends the keyframe to the backend. Additionally all the information registered for all the frames since the last keyframe is also sent to the backend. The pieces of information included are the frame poses and all the instances of people detections.

The backend generates the human activity semantic layer, composed paths of each person with respect to the unpopulated static map, irrespective of where the camera was when the people were detected. These trajectories are computed by the composition of the 3D position of each person with respect to the camera with the 3D camera pose computed by the frontend. After each iteration of the backend, the position of each keyframe is updated. We keep the connection of the intermediate frames with the keyframes to propagate the position updates of the keyframes to the intermediate frames, and from there to each person's trajectory, which is updated at same rate as the unpopulated map. This

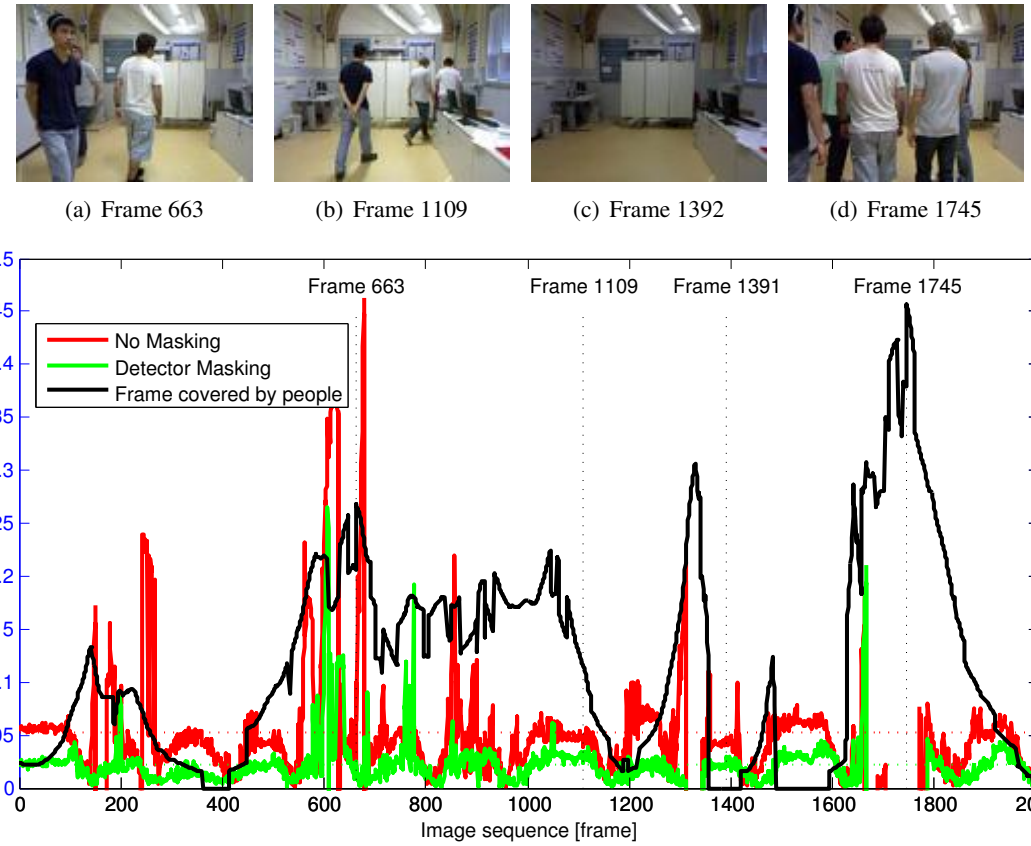


Figure 5.4.: Effect of human activity masking per each frame of the sequence "Translation" processed by ORBSLAM2. Top row displays a representative case.

information adds a semantic label to activity layer of the map as a "occupied area" by people. This semantic information will be use by a mobile robot for navigating in the same environment.

5.4. Experiments

In this section, we present the experiments performed for validating our approach. The system is implemented in C++ using Robot Operating System (ROS) Quigley et al. (2009). An Intel Core i7 @2.67GHz processor was used for running the VSLAM system and the people detector at a 24 fps rate. For validating the approach we have used the Kinect Tracking Precision Dataset (KTP) presented in Munaro et al. (2016). This contains 5 different sequences acquired with a Microsoft Kinect on board a mobile platform. We have selected this dataset because it provides RGB-D images in a context where a robot makes a trajectory and also includes a ground-truth of both the position of the robot and the detections of the people. Regarding the VSLAM system, we have tested our approach using two different PTAM-like algorithms adapted for RGB-D, C^2TAM Riazuelo et al. (2014b) and ORBSLAM2 Mur-Artal and Tardos (2016). Both of them use Jafari et al. (2014) for human activity detection.

Table 5.1 presents the results of both VSLAM algorithms running over two of the KTP sequences:

	Translation Sequence		Arc Sequence	
	ATE (mm)	% Frames Relocated	ATE (mm)	% Frames Relocated
NoMasking	51.2	87.5	32.7	78.8
Ground-Truth Masking	16.2	76.8	21.6	56.7
People Detector Masking	17.8	75.2	22.5	37.2

Table 5.2.: Camera relocation performance over KTP dataset sequences using ORBSLAM2.

"Translation" and "Arc". The metric selected for measuring the performance of the system is the Absolute Trajectory Error (ATE) after aligning the computed trajectories with the ground truth by means of a rotation and translation. We also include the rate of frames successfully located, named "% Frames Tracked". We run the sequences with three different configurations for human activity detection: *No masking*, in which human activity detection and masking is deactivated; *Ground-Truth Masking*, where we use the ground truth bounding boxes detections provided by KTP, in order to find an upper performance limit; and *People Detector Masking* by which the system is fed with the real people detection provided by Jafari et al. (2014).

First of all we can see how the frame tracked percentage is very similar for all the configurations. However the ATE error is reduced by more than half if human activity masking is applied to the images, which proves the benefit of masking. Also we can see a small increase in error depending on whether we use the ground-truth detections or the people detector. This increase in the error is minimal, proving that we are close to the upper performance limit.

Figure 5.4 presents in detail the evolution of the error during the whole sequence for the "Translation" and ORBSLAM2 case. The plot displays the error with people detector masking (green line) and with no masking (red line). The area of the image covered by people is also plotted, as an index of the human activity. The camera location error is highly correlated with the level of human activity. Four representative frames are selected in order to visualize this correlation. In the frames 5.4(a) (#663) and 5.4(d) (#1745) we can see how the error increases due to the increment in human activity. The higher the occlusion level of the image caused by the people in the scene, the lower the system accuracy. There are even situations in which the occlusion level is so high (Frame 5.4(d) (#1745)) that the system cannot be located and the camera is lost. In addition we can see in frame 5.4(b) (#1109) how the human activity level decreases and hence the error also decreases. When there are no people in the image (Frame 5.4(c) (#1391)) we can see in the figure how the error is maintained or even decreases.

For validating the camera relocation performance we have run KTP dataset "Translation" and "Arc" sequences over ORBSLAM2 because its relocation algorithm based on bag of words is on top of the state of the art. We have run the VSLAM system on the first part of the scene (about 300 frames) during the scene exploration stage. Afterwards we close the map and force a camera relocation for each frame in the rest of the sequence. Table 5.2 presents the results of the camera relocation performance for different options of masking, applied both to the map creation and the subsequent relocation. We can see the improvement in the ATE metric when human activity is masked. Regarding the use of the ground-truth for masking the people, the ATE metric is quite similar to that achieved by the implemented detector. We can also see how the percentage of relocated frames drop with people masking, this is due to with no masking some frames are not relocated correctly and the number of relocated frames increases.

Figure 5.5 presents the evolution of the relocation error with (green line) and without (red line)

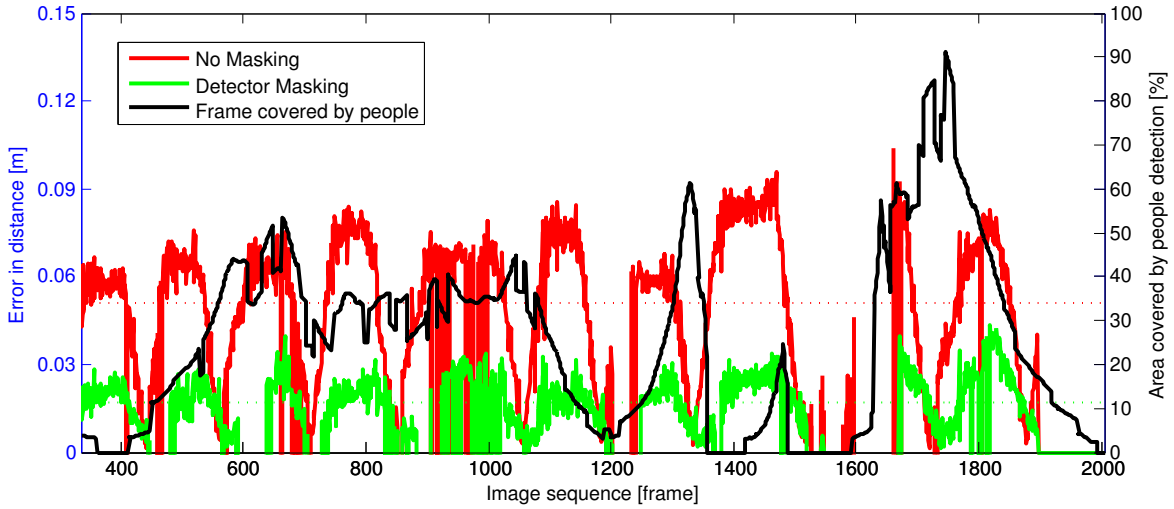
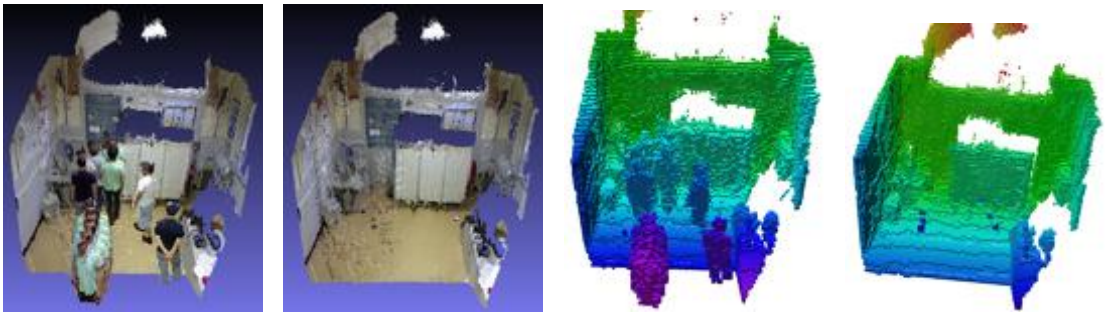


Figure 5.5.: Effect of human activity masking on relocation. ATE after relocation per each frame of the "Translation" sequence processed by ORBSLAM2.



(a) (b) (c) (d)

Figure 5.6.: 3D reconstruction and Octomap using unmasked 3D point cloud (a,c), and using the human activity masked data (b,d).

masking people. In this case we can also see the correlation between the error and the area of the image covered by people. In the frames in which the image is covered by people, the error is greater than in which are without people. In addition, depending on the level of occlusion, the relocation algorithm is unable to provide a position. We can conclude that using a masking technique improve the camera relocation performance.

Regarding the geometrical layer, figure 5.6 displays the improvement provided by our C^{2}TAM approach after the initial exploration of the scene ($\approx$ 350 first frames). Due to the information provided by the people detector about the estimated position of the people in the images, a process of annotation and removing the people from the 3D point cloud is performed. Figure 5.6(b) shows the result of this process. The number of points in the scene is reduced in a 30%, and there is not any evidence of people in the scene, in contrast to raw the reconstruction shown in figure 5.6(a) where artifacts due to people are noticeable.

Comparing the initial texture alignment with the octomap approach, we can see in the figure

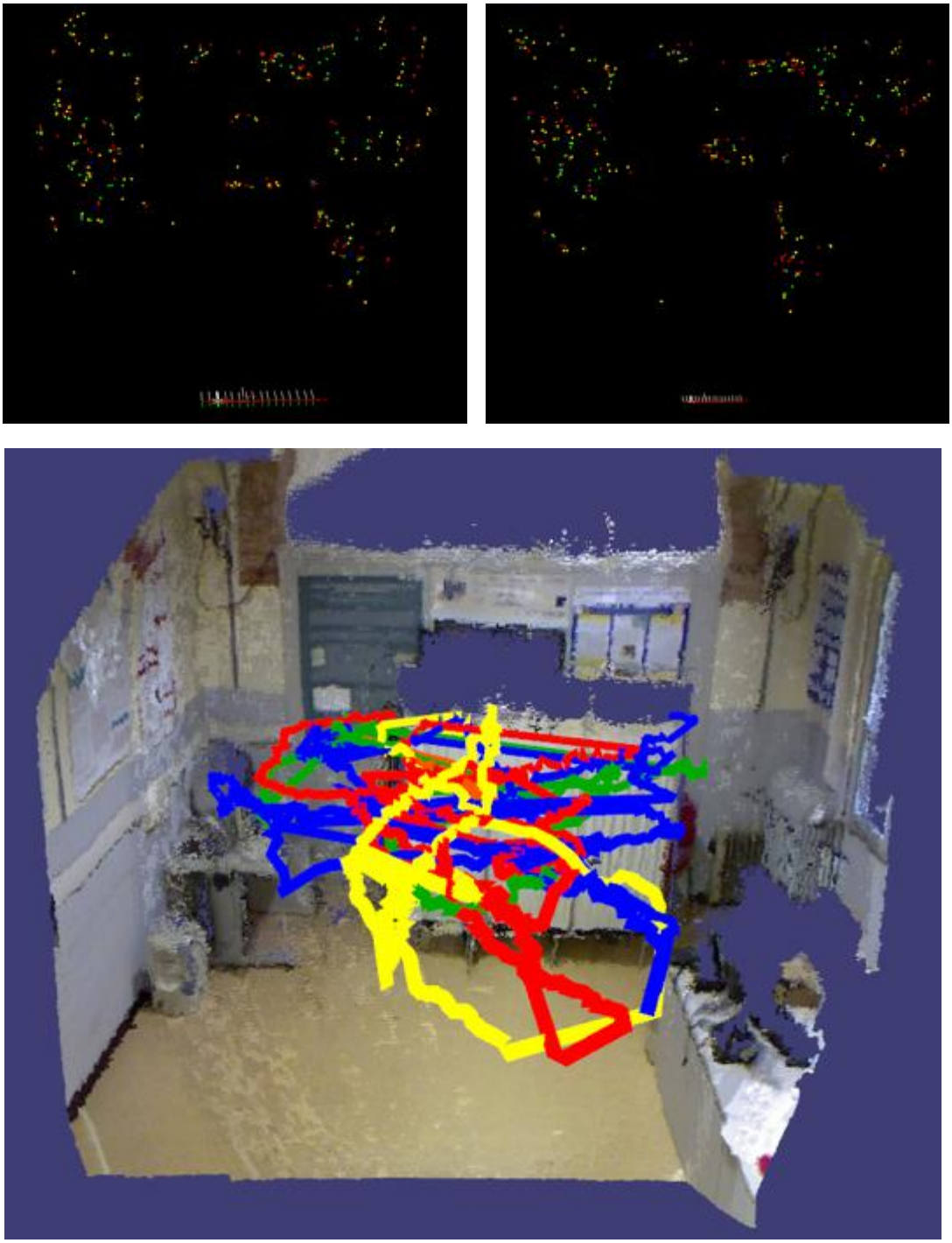


Figure 5.7.: Sparse map and camera trajectory (top), and geometrical and human activity layer (bottom).

the maps obtained. In figure 5.6(c), an octomap is built using the raw data from the 3D point cloud associated to each keyframe witout masking. In contrast, using our approach (figure 5.6(c)), the octomap built masking the people information does not even include static persons.

Figure 5.7 summarizes the results obtained by the VSLAM system proposed. Top images show the sparse map (front and top view) generated by C^2TAM after masking. The map is used for locating the RGBD camera in the scene. We can see the feature points extracted from the rigid scene and the camera trajectory followed by the robot during the experiment. In the bottom image, we can also see the unpopulated map that describes the 3D geometry where information corresponding to the people has been removed. In addition, the people trajectories that define the semantic layer are also displayed. A full video of the validation can be found in *.

5.5. Discussion

Thanks to the embedding of human detection in a rigid visual SLAM system, it is possible to build precise VSLAM maps in populated environments.

Our system integration approach can effectively remove human activity from an unpopulated map simply by removing the people from the input sequence. People detection and removal not only benefits the scene description, it also benefits the camera tracking and relocation, significantly improving their performance in populated environments, and reducing the ATE error by more than half. Additionally it is possible to generate a semantic layer that exhaustively registers human activity in the mapped area, irrespective of the camera motion during acquisition. We can conclude that both the mapping and the people tracking benefits from the combination. In addition, the system is able to deal with the false positives provided by the person detector, since the detection is performed in all frames of the sequence.

We have tested the embedding in two popular rigid VSLAM systems, with a similar gain in performance. We can therefore conclude that our proposal is agnostic to the VSLAM method, and hence can potentially improve any SLAM system. In the same way, we can say that the combination of this method map building with the recognition of people is agnostic to the algorithms used, since any detector that give us information of the person in the scene can be included in the proposed system.

People detection can be considered as an instance of class category recognition and tracking. A similar benefit in terms of an increase in the robustness, map reuse and tracking of the instances irrespective of the camera motions could be achieved in other environments, for example vehicle detection for automated driving, or a tool detector for endoscope-guided minimally invasive surgery.

Mapping populated environments where human movements are prevalent is a key capability for service robots. As future work we plan to develop robot motion planners in such environments, profiting from the human activity mapped. In our opinion, planning and navigation using this information can be improved by the knowledge of the patterns of human movement.

*https://youtu.be/HKe1-kXtM_E

Chapter 6

Service robotics in confined and structured environments

6.1. Introduction

In previous chapters it has been presented some approaches to allow a service robot to understand the environment by incorporating semantic information (”objects” and ”people”). These works have integrated visual SLAM systems and object and people detectors. The importance of semantics for map building has been demonstrated. This chapter will deal with semantic concepts similar to those previously applied in the field of safety, security and rescue robotics (SSRR). We present a part of the work published in Tardioli et al. (2016). Our contribution have been focused on: 1) distributed map building; 2) semantic navigation based on place recognition; 3) integration of the different modules into a complete system; 4) multiple experiments in real-world scenarios; 5) evaluation of the experimental results and extraction of conclusions and lessons learned.

The work performed aims to deal with intervention tasks in confined and structured environments, as a previous but necessary step to dealing with more complex scenarios. These environments poses multiple challenges that involve task planning, motion planning, localization and mapping, safe navigation, coordination and communications. We propose a complete framework which faces the aspects and issues that can appear in this kind of scenarios with the aim of successfully completing an intervention mission. The goal is to carry out a supervised deployment of a robot team in a way in which they autonomously reach a goal and give to the human operator the possibility of teleoperating one of them to explore the environment.

The contributions presented in this chapter and hence integrated into the framework are: i) multi-robot autonomous motion planning and navigation combined with teleoperation capabilities, mandatory in most robotic missions; ii) semantic and topological localization combined with geometric localization and mapping; iii) remote map building using lidar sensor data. From the point of view of the human team supervising the mission, it is important that the deployment plan is described in an easy and flexible way. To do this, the mission plan is transmitted to the system in terms of achieving several high level topological-semantic features that the perception system of a leader robot exploring the environment is able to recognise and locate. Similar to action recipes presented in chapter 4 in which semantic wa related to objects to be recognized and located in the scenario; in this chapter the semantic is related to relevant places to recognize to locate and navigate.

We develop a complete robotic framework —composed of a set of mobile robots and a base

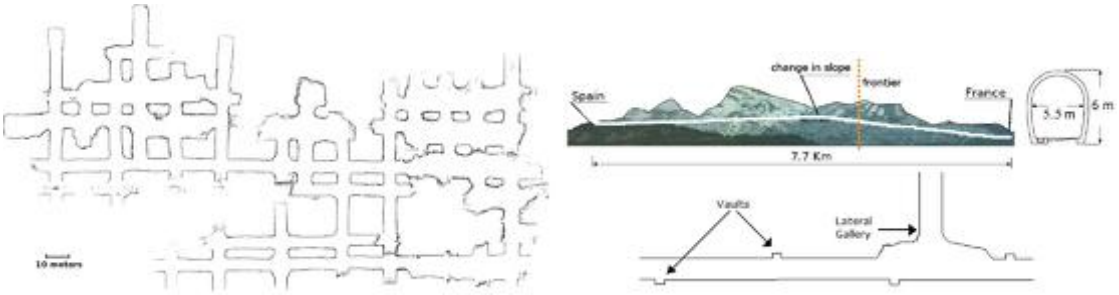


Figure 6.1.: Partial map of the Santa Marta mine. Courtesy of Minera Santa Marta S.A. (left). Somport tunnel and small vaults and lateral galleries (right).

control station— as a proof of concept for intervention in structured and large environments. In figure 6.1 a snapshot of each of the scenarios chosen to develop the experiments are shown. The mission to be achieved is reaching a location without relying necessarily on a previously known metric map. Both scenarios will be used for the evaluation of the system proposed in this work. We assume that the only information available to the system is semantic and topological, and expressed in terms of features or natural landmarks for navigation (e.g. left or right gallery, cross intersection, corridor, etc.). In these scenarios it is not usual to have a communication infrastructure available, or else it has been destroyed. The goal is to provide a complete system for allowing a team of robots to reach the destination under these constraints and limitations and enable the operator to teleoperate one of the robots —that we call the leader— at a desired area. These scenarios allow an assessment of the performance of each of the functionalities of the system and of the system as a whole for achieving the mission.

The first scenario is a large mine built as a maze of galleries, in which a real mission can be to transport materials or tools to several locations. This environment is challenging because autonomous navigation and self-localization of the robots demands capabilities to recognize the places and localize themselves, whilst the mission plan is described by the supervisor in terms of semantic features representing the places. This scenario serves mainly to evaluate: i) the robot deployment planning algorithm, based on semantic-topological features perceived by the sensors onboard; ii) the robust recognition algorithm for those features from their semantic meaning, and their topological localization in the whole scenario. Although the scenario is real, obtained by manually navigating a robot, the system is evaluated by simulations in order to use more robots and a complex environment.

The second scenario is a large railway tunnel, in which there is a long corridor and multiple lateral galleries. It is simpler than the first scenario, from the point of view of navigation, so it is used mainly to evaluate the communications and network issues in real experiments. In this scenario, two experiments were carried out: the first one to evaluate the whole system and all the functionalities together including teleoperation in a location far away from the base station and the associated real-time aspects. The second one was carried out in a part of the tunnel, with the objective of assessing the robustness of the system to be able to replan the robot deployment when unexpected situations occur such as a blocked pass or loss of communication.

The rest of the chapter is organised as follows: Section 6.2 refers the related work. Section 6.3 presents an overview of the whole system and section 6.4 introduces the deployment algorith. Finally, section 6.5 shows the experimental results and section 6.7 concludes and presents the lines for the future work.

6.2. Related Work

Several applications and projects involving safety and rescue in tunnels using robots have been developed in recent years. These include robots for fire-fighting, exploration, inspection, sensor deployment or surveillance. In Murphy et al. (2009) and Habib et al. (2011), the authors offer a survey on challenges in robots deployments in subterranean and harsh scenarios. Autonomous navigation, localization, mapping and communication are presented as basic capabilities needed in order to accomplish a mission in such environments. However, in most of the works relating to applications in confined environments appearing in the literature, these kinds of challenges are only addressed in isolation.

In Walker (2009) the author describes a project involving a robot for surveillance in underground galleries against smugglers in national borders. In White et al. (2010), an underwater robot for archaeological teleoperated exploration in cisterns is presented. The work focuses mainly on SLAM (Simultaneous Localization and Mapping) problems for mapping the cisterns. In Zhuang et al. (2008), a robot for inspecting tunnels, capturing images and registering the concentration of some poisonous gases is developed. In Zlot and Bosse (2014) a laser based 3D SLAM solution is presented for mapping large scale underground environments such as mines. All these works, however are focused only on a partial view of the whole problem: mechanical aspects, SLAM, or solving particular aspects of the application. None of them propose a complete framework for robot operations in confined environments.

During the last few years some works have attempted to build topological maps for robot mapping and navigation. Concerning appearance methods, some of them have been applied for place recognition Mozos et al. (2007), for SLAM at a topological level Newman et al. (2006) Cummins and Newman (2008) and for navigation purposes Booij et al. (2007). Granström et al. (2011) describes a technique for closing loops extracting geometrical and statistical properties from point clouds of range data. They apply appearance similarity for matching places but no semantic sense is provided to the features. In Thrun and Montemerlo (2006) and Grisetti et al. (2010) Graph SLAM algorithms are proposed. In Ranganathan and Dellaert (2011) the authors develop a probabilistic topological mapping, and in Tully et al. (2009) a pure topological graph is described for closing loops. It is improved in Tully et al. (2012) by developing a probabilistic framework that builds hybrid metric-topological maps. Marinakis and Dudek (2010) propose a method for exploration strategies building a hypothesis tree and focusing on reducing the number of hypotheses using mainly topological rules. However, in these papers no semantic interpretation is associated to the measurements. In Kuipers et al. (2004); Zender et al. (2008) the authors propose appearance methods for building hybrid metric-topological maps augmented with semantic information, but they do not use a stochastic framework at a topological level.

Multi-robot SLAM algorithms, Burgard et al. (2000), Howard (2006), Lazaro et al. (2012), have been proposed to integrate the partial views of the robots and to build a more complete view of the environment in a common map, in which to localize and plan motions for the team. These techniques improve the localization, mapping and planning processes, although they increase the computational requirements, mainly for large environments.

In Peasgood et al. (2006) a work on multi-robot path planning in tunnel-like scenarios is presented. However, only simulation results are shown, and the multi-robot communication problems are not addressed. In Bakambu and Polotski (2007) an underground scenario similar to those used in this work is presented. Laser rangefinder information is used to recognize some characteristic places such as corridor intersections and to autonomously navigate in this scenario. An ad-hoc technique is developed for this semantic recognition, and a path planner computes actions to navigate among the recognised natural landmarks. In the mission carried out in a real underground scenario, com-

munication and connectivity problems are not addressed. However, communication capability plays an important role especially in applications that involve multiple agents, when teleoperation capabilities are required or when the transmission of significant amount of heterogeneous data (such as video, maps, goal points, victim data, and so on) between the robots and the mission control base is necessary Birk et al. (2009).

In the new century, wireless communication among mobile robots has attracted a growing interest. The main concept is to navigate acting on the robot motion to maintain the communication between robots and with the base station. Nguyen et al. (2004) proposes, in this seminal work, the use of a set of slave robots acting only as communication relays, to guarantee the link between a teleoperated robot and a base station. In Dixon and Frew (2009), Tardioli et al. (2010), and Sabattini et al. (2013) a set of fully functional robots perform the missions constrained by a reaction mechanism, based on sensing the communication link quality, which ensures the network connectivity.

6.2.1. Challenges

An application of this magnitude, involves a very extensive set of tasks and activities which include the following: i) robot motion planning and navigation, ii) topological and semantic interpretation of the environment, iii) localization and mapping, and iv) real-time and multi-hop communication. Each task, per se objectively difficult, is made even more complicated in the hostile environments where the system is planned to be used. In the following sections we explain the obstacles each of these tasks must overcome.

Robot motion planning and navigation

Robust motion planning and obstacle avoidance are basic requirements for autonomous behaviour, even when the robots are teleoperated. In fact, the system must be robust against a possible temporary failure of the localization algorithm or the communication. This means that navigation must integrate both global and local planning based on local information. This poses a challenge as to how to autonomously navigate in large environments for which there is not a precise geometric map or which is built while the robots navigate, without crashing against obstacles or lateral walls. A solution is to use topological-semantic information, which avoids to manage a lot of geometric information for accurate localization, which in many cases is not needed to achieve a mission.

walls or centred in the corridor and in the galleries, whilst avoiding obstacles. A lower level obstacle avoidance algorithm, the ORM-HS introduced in Rizzo et al. (2013), guarantees that the robot will not crash in the case of dynamic or static obstacles. The choice of this algorithm was due to the fact that it is based on the well known ND algorithm Minguez and Montano (2004) that is capable of navigating effectively in dense and difficult scenarios (such as sites of accident or collapse), but improved for cluttered and uncluttered environments, such as those selected here.

Semantic and topological interpretation of the environment

In large environments with partial, incomplete, or imprecise previous information or maps, it is not possible to achieve a mission describing the goals and subgoals in geometric terms, especially if any kind of exploration is needed. When the place where the exploration must be carried out is remote, far away or simply difficult to reach, it can be complicated and dangerous to teleoperate the leader robot up to such as position. There is, in fact, the possibility of human error, due to the necessity of maintaining a high level of concentration during long periods. This can delay the intervention

and even jeopardize the success of the operation. To avoid this, the proposed scheme allows the user to specify the mission steps in terms of topological features or natural landmarks. The feature detection, together with the autonomous navigation and topological localization, alleviates the work of the human operator who only needs to supervise the team deployment until the destination location is reached. To do this, we here propose a technique to recognize topological features having a semantic meaning based on preliminary work, Romeo and Montano (2006). The systems learns several features from artificial data sets, but it generalizes the recognition to others which are similar although not equal to those that have been learned.

Localization and map building

While the leader robot moves towards the goal following the mission plan described in terms of topological features, a map can be built from the information coming from its perception system. This map can be used for the localization of the rest of the robots (the followers), and to obtain a more precise description of the scenario. Geometric maps either based on features such as segments, corners, etc., or else grid maps, can be built.

Sharing and integrating different views of the robots in a team to build more complete maps is another possible appproach. Multi-robot SLAM algorithms as Burgard et al. (2000), Howard (2006), which integrate information from several robots could be used. However, in the type of scenarios and applications considered in this work, this kind of approach does not provide additional relevant information about the environment —all the robots navigate in the same environment and even following the same path— but it would increase the real-time computational burden.

A SLAM module has been included in the system, responsible for simultaneously localizing the leader robot and building the map of the environment using its laser readings. Due to the nature of these environments, it is best to use laser measurement sensors because lighting may be poor and visual sensors do not provide useful information. As the objective of the present work is not specifically oriented to developing new SLAM techniques, we have chosen a well-tested state-of-the-art SLAM algorithm Grisetti et al. (2010) that provides very accurate results in dense and featured environments. The follower robots are localized using a state-of-the-art particle filter technique Fox (2003) over the map generated by the leader robot provided online to the localization algorithm, when needed. The localization of the follower robots on the map is used by the robot deployment planning algorithm to locate these robots in the positions in which the communication with the base station throughout network is always maintained. Other approaches to mapping based on features instead of on grid maps Lazaro et al. (2012) could be applied in order to reduce the need for storing and managing large amounts of information, but these lie outside the scope of this work.

Real-time and multi-hop communication

One of major issues in this kind of applications and environments is the communication among robots. There are clear requirements for firm real-time and multi-hop communication given that the perception-actuation loop includes the exchange of information between the leader node and the base station. Thus, we decided to adopt the RT-WMP protocol Tardioli et al. (2014) that provides the communication support required by the system: multi-hop communication, fixed priorities of messages, bounded delays, link quality information and mobility management. The link between robots is assured by a set of relay robots strategically deployed that must overcome a set of communication problems such as distance path-loss, fadings and abrupt signal vanishing at corners. This aspect has previously been discussed in greater depth in the works presented in Tardioli et al. (2010) and Rizzo

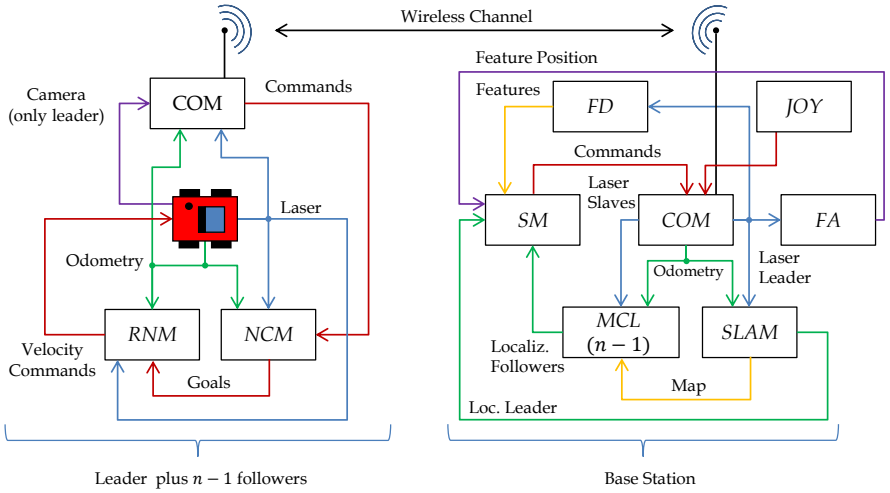


Figure 6.2.: Conceptual architecture of the system. The different modules are: COmmunication Module (COM), Reactive Navigation Module (RNM), Navigation Control Module (NCM), Monte Carlo Localization (MCL), Simultaneous Localization And Mapping (SLAM), Feature Analyzer (FA), Feature Detector module (FD), State Machine module (SM), and teleoperation module (JOY).

et al. (2013).

6.3. System Description

This section provides a brief overview of the whole system and a detailed description of the system components.

6.3.1. Overview

The system has been organized into several interconnected modules. Figure 6.2 shows its conceptual structure. It is composed of a set of n mobile robots, one of which will assume the role of leader and $n-1$ followers. The leader and the follower robots have very basic characteristics: the idea is that all of them can be low cost devices that can be replaced easily or even be disposable. They are built on a basic platform that provides LIDAR and odometry readings, and that is capable of receiving velocity commands. From the conceptual point of view, they are equipped with three modules: the Reactive Navigation Module (RNM) which manages the collision avoidance, the Navigation Control Module (NCM) which computes the trajectories, and the COmmunication Module (COM) which connects all the nodes of the network. The robots mostly executes the commands that they receive from the base station through the COM. These commands are translated by the NCM into one of the two predefined navigation strategies: the first consists of the capability of reaching a goal provided by the base station, and the second of navigating forwards following a lateral wall or centred in corridors. In both cases the RNM translates the goals into velocity commands compatible with the avoidance of potential obstacles. The leader also has a camera that, when active, sends images to the base station through the COM.

The base station is responsible for carrying out most of the tasks. It is composed of several

modules that interpret the scenario through the recognition of semantic features, localizes the robots, build the map, plans the goals, and manages the communication. The SLAM module receives laser and localization data from the leader through the COM and, simultaneously, localizes it and builds the map of the environment in real-time. This map is provided with a certain non-constant rate to the $n-1$ MCL (Monte Carlo Localization) modules that, together with laser and odometry data from the followers, localize these latter. This enables all the robots to be localized over the same map, built on the fly. The Feature Detector module (FD) receives the laser data from the leader and detects the different semantic characteristics of the environment (left or right gallery, end of corridor, etc.), and interprets them in order to topologically localize it. Also, the Feature Analyzer (FA) module analyzes the data coming from the leader laser and identifies the goal position inside the feature, when needed. All this information is provided, together with the localization of all the robots coming from the SLAM and MCL modules, to the State Machine module (SM) that is responsible for making all the decisions for the correct execution of the plan. Finally, the JOY module generates the commands that allow the teleoperation of the robot when a manual control is requested. In the following section the different modules will are explained in more detail.

6.3.2. Localization and Map Building

The localization of the leader and of the followers relies on a state-of-the-art SLAM algorithm and on a Monte Carlo localization approach respectively. As the system does not have a known geometric map, the leader explores the environment by navigating in an unknown scenario, searching for the semantic features described in the high level plan, as explained in detail in section 6.3.4. Simultaneously it uses a Rao-Blackwellized particle filter approach Grisetti et al. (2005, 2007) to compute an accurate position taking into account the robot movement and the most recent observation. The followers are localized on the map built by the leader by means of a probabilistic localization using the Augmented Monte Carlo Localization (AMCL). As described by Fox (2003), it implements the adaptive KLD-sampling Monte Carlo localization approach which uses a particle filter to track the pose of a robot against a known map. The idea behind KLD-sampling is to determine the number of particles based on a statistical bound on the sample-based approximation quality. Both algorithms are those ones developed in the ROS platform.

Both units are fed with the laser data and the odometry estimation coming from the leader and the follower robots through the COM module. The map generated by the SLAM is provided to the AMCL units. However, since the reinitialization of the particle filter with a new map is a quite costly task, this takes place only when it is strictly necessary. This means that the map is provided individually to the follower robots only when they need it for localizing themselves to reach a goal. Specifically, the state machine managing the system forces the updating of the map in the AMCL units just before requesting the corresponding follower to move.

6.3.3. Navigation and obstacle avoidance

As mentioned above, the robots have to explore the scenario by navigating in an unknown environment. For this purpose, it is necessary to provide obstacle avoidance navigation in order to safely move in such large environments. The RNM module provides this. This module uses the ORM-HS which extends the Obstacle-Restriction Method (ORM) Minguez (2005) based on the Nearness Diagram family methods Minguez and Montano (2004); Minguez et al. (2001). Due to the nature of the experimental environment, the use of the ORM-HS method provides a high performance in cluttered environments and also provides the capability to increase the velocities of the robot in a non-dense

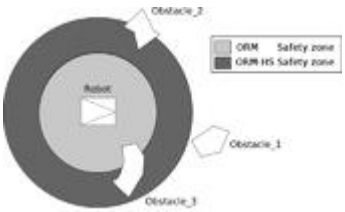


Figure 6.3.: Diagram of the safety zones.

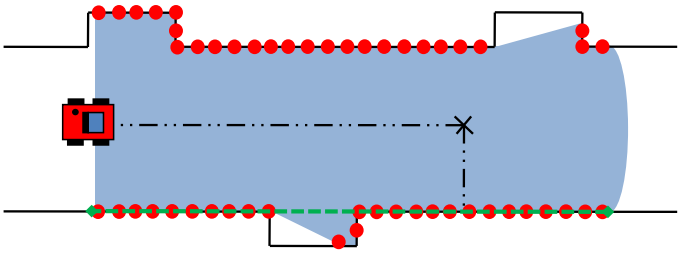


Figure 6.4.: Data sensor segmentation and autonomous goal computation.

scenario. It is essential that the robot is able to navigate at the highest speed possible in environments where there are no obstacles because the distances involved in the experiments that must be traversed by the robots are large. Figure 6.3 presents a drawing that summarizes the safety zones of the obstacle avoidance method. If there are no obstacles inside the ORM-HS safety zone (Obstacle_1) the robot navigates at maximum speed. When an obstacle enters into the zone (Obstacle_2) the velocity is decreased proportionally to the distance to the obstacle. Finally if there is an obstacle inside the ORM safety zone (Obstacle_3) the robot starts to navigate in a cluttered environment reducing its speed.

The Navigation Control Module (NCM) is responsible for dynamically translating global goals to the local frame of the robot. Also, it includes several navigation strategies depending on the task to be executed. The following paragraphs summarize the different navigation methods included in the NCM module.

Autonomous navigation

The autonomous navigation modes has been conditioned by the nature of the environment. As the robots have to navigate in a huge environment (sometimes of several kilometres in length) and far from the human supervisor of the mission, it is necessary to provide the robot with autonomous navigation methods that do not require interaction with the base station and limits the operator's responsibility for this task. This way the robots can navigate during long periods until reaching the objective, without external control.

There are different autonomous navigation methods depending on the role of the robot. The first has been designed for the exploration of an unknown environment and it is performed by the leader. The leader is responsible for providing LIDAR sensor data to the base station in order to build a map that will be used by the rest of the robots, as mentioned previously. Using a laser data segmentation method based on the Hough transform, this module is capable of obtaining the lines corresponding to the straight walls of the tunnel, as shown in figure 6.4 (more details can be found in Tardioli and Villarroel (2014)). These segments are used for computing a goal parallel to and at a fixed distance from the wall, and from the robot. This goal is sent to the RNM module that responsible for carrying out the reactive navigation towards the goal which, incidentally, is re-computed continuously at the same sensor frequency. The second method has been designed for the follower robots that, in contrast to the leader, can rely on the map that the latter build on the fly. Thus, these are continuously fed with global goals referred to the map frame that are then converted to local goals by the NCM.

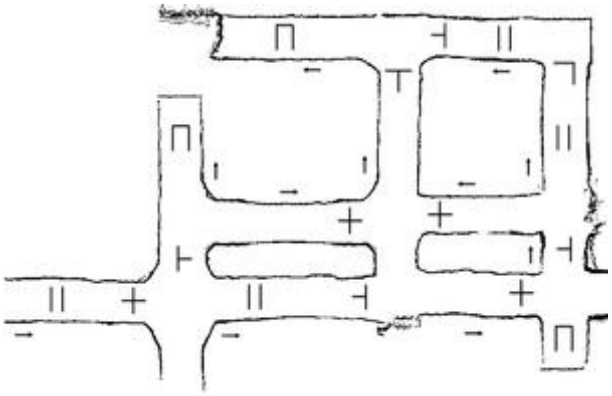


Figure 6.5.: Example of scenario and its associated topological map. The symbols in the nodes represent some of the different features that can be recognized. Their meaning is defined in Table 6.6.

Perceptual feature	Symbol
Diaphanous corridor	‖
End of corridor	⊓
Cross	+
Left hand	⊣
T-intersection	⊤
Right hand	⊢
Left turn	¬
Right turn	⌜
Unknown	?

Figure 6.6.: Semantic features learned for topological localization.

6.3.4. Recognition of Semantic Features

The proposed appearance-based recognition system is based on the work presented in Romeo and Montano (2006). It is capable of learning different semantic features typically present in indoor maze-like environments. Figure 6.5 shows an example of applying the technique to a part of the mine scenario. The method retains the fundamental information about the feature ”shape”, being capable of learning features not strictly polygonal (curves, non planar walls, intersections with relative angles different from 90 degrees, etc.) or partially disturbed by people moving around. The semantic features to be recognized are the typical features found in maze-like environments. Table 6.6 represents all the selected features to be learned and their symbols. These are features that can be obtained from a 2D laser rangefinder sensor, but could also be applied to information computed from 3D range or RGBD sensors. The system was trained from artificial sets of environments, but is applied to others and to real ones, not previously learned. In this work use an extended version of the algorithm proposed in Romeo and Montano (2006), improving the learning algorithm so that it is more robust and discriminatory for different environments.

Learning procedure

The semantic features to be recognized are the typical features found in maze-like environments 6.6. These are features that can be obtained from a 2D laser rangefinder sensor, but could also be applied to information computed from 3D range or RGBD sensors. The method first performs an input space (laser scans) transformation based on *Principal Component Analysis* (PCA), and then a *Linear Discriminant Analysis* (LDA). This process captures the main characteristics of the features and reduces the information in order to obtain a clear and robust discrimination. Once the input space has been transformed, a bayesian classification scheme is applied in order to recognize the features. The posterior probability will be used not only for obtaining the most likely feature, but as a metric for the reliability of the selected feature with respect to the other possible features to be recognized. A probability above a threshold will allow the feature to be accepted or rejected. Using the same classifier jointly for all the features yields a not very robust discrimination. This conclusion was obtained from

several experiments. In this work we include an improved version of Romeo and Montano (2006) algorithm focused on the discrimination capability in two directions:

- By performing a non linear classification scheme based on *Kernel Fisher Discriminant Analysis*.
- By increasing the specialization of the classification procedure by using a two step method.

Non-linear discriminant analysis

Kernel Fisher Discriminant Analysis (KFD) Baudat and Anouar (2000) provides classification capability for nonlinear problems. The basic idea of KFD is to solve the problem of LDA in the feature space $\mathcal{F}$, thereby yielding a set of nonlinear discriminant vectors in the input space. It can be reformulated in an easier (and more understandable) form, as a two-phase process: *Kernel Principal Component Analysis* (KPCA) plus LDA Yang et al. (2004).

Classifier specialization

In order to improve the discrimination capacity of the feature recognizer, the algorithm used increase the specialization of the classification procedure by using a two step method. During the first step, it will discriminate between *open* and *closed* features based on the first component value. Later, in a second phase, the method will apply either specific equations using specialized matrices, according to the result of the first step. Specialized matrices have been obtained applying the analysis to the corresponding subset of features:

- $\|$, $+$, $\dashv$, and $\vdash$ for *open* features.
- $\sqcap$, $\top$, $\urcorner$ and $\ulcorner$ for *closed* features.

It is worth mentioning that this is a general method which learns from generic artificially created features but is capable of recognizing those kinds of features present in not-previously-visited real and noisy environments. Its performance is evaluated in the experimental results section 6.5.1.

6.3.5. Communication module

The communication module is based on the RT-WMP protocol Tardioli et al. (2014). This is a protocol for MANETs that works on top of the 802.11 protocol and supports firm real-time traffic. Moreover, the protocol provides global static message priorities and supports multi-hop communications. The target application for this protocol is that of interconnecting a relatively small fixed-size group of mobile nodes, generally mobile robots (up to 32 units). It is based on a token passing scheme and it is designed to manage rapid topology changes. The routing algorithm of the RT-WMP is based on the link quality among nodes: to describe the topology of the network, RT-WMP defines a network connectivity graph having non-negative values on the edges. These values are computed as a function of the *Radio Signal Strength* (RSSI) between pairs of nodes and are indicators of *link quality* between them. They are stored in the so called Link Quality Matrix (LQM). Figure 6.7 shows the RSSI measurements in the Somport tunnel as a function of the position.

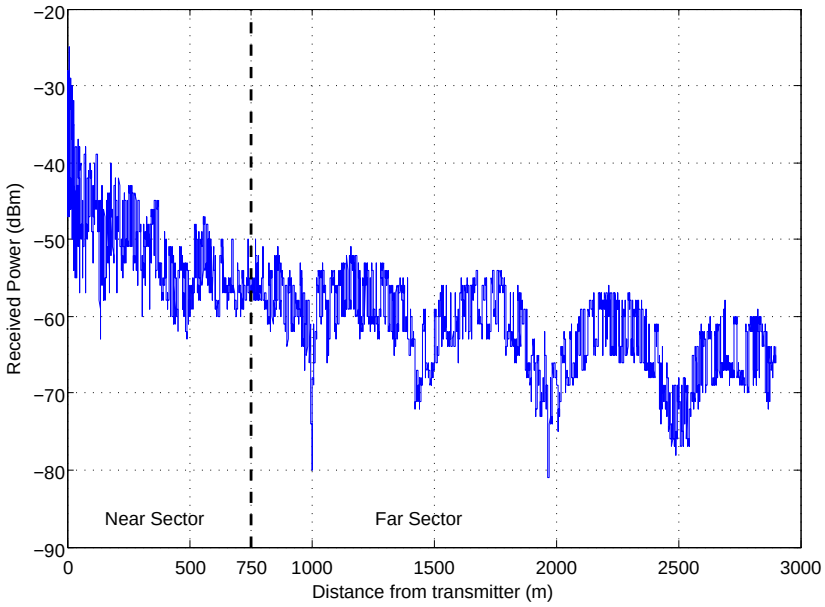


Figure 6.7.: Measured received power (dBm).

6.4. Deployment Planning and Navigation

The algorithm is ruled by a simple state machine which, interpreting all the data coming from the leader and follower robots and taking into account the radio link quality among the nodes, makes the decisions for the development of the mission. Figure 6.8 shows an intuitive graphic deployment of the algorithm. A simplified deployment algorithm for an n-robot team is shown in Algorithm 6.4.

At the beginning the $state$ is fixed to $BEGIN$, the plan is read, and the variable $last\_mobile$ —that contains the identification of the last mobile node— is set to $n-1$ which represents the last mobile node of the chain (lines 1-3). Also, the variable id, an index that points to the current feature to be found according to the plan, is set to 0.

Then, the robots are virtually roped in a chain $R_0 \leftrightarrow R_{n-1} \leftrightarrow R_i \leftrightarrow R_2 \leftrightarrow R_1$ (lines 4, 6) in order to create a mechanism that guarantees the connection of the network at all times. The rope is represented by the radio signal sensed by each of the nodes and that the state machine accesses through the shared LQM that circulates in the network. More details can be found in Tardioli and Villarroel (2007). There exist two thresholds: *upper* and *lower*. When the radio signal between two robots falls below the first threshold —e.g. when R_1 moving away from R_2— the rope will provoke R_2 to start moving to recover a safe signal level. If, despite the movement, the signal continues to fall or falls abruptly —for example if for some reason the back robot (R_2 in the previous example) can not move or an obstacle appears between the two of them delaying the movement— and reaches the lower threshold, the forward robot (R_1 in the example) is stopped. It will be kept still until the signal quality with the back robot reaches the *upper* threshold again.

An exception to this behaviour is the end of the chain since robot R_{n-1} cannot force the movement of R_0 given that, being the base station, the latter is fixed and immovable. This guarantees that, if the robots reach the maximum elongation of the virtual ropes, all of them —one after the other— will be stopped to preserve the network integrity. When, during the development of the mission, the hindmost mobile robot is stopped, the variable $last\_mobile$ is decreased by one. The mission has failed when

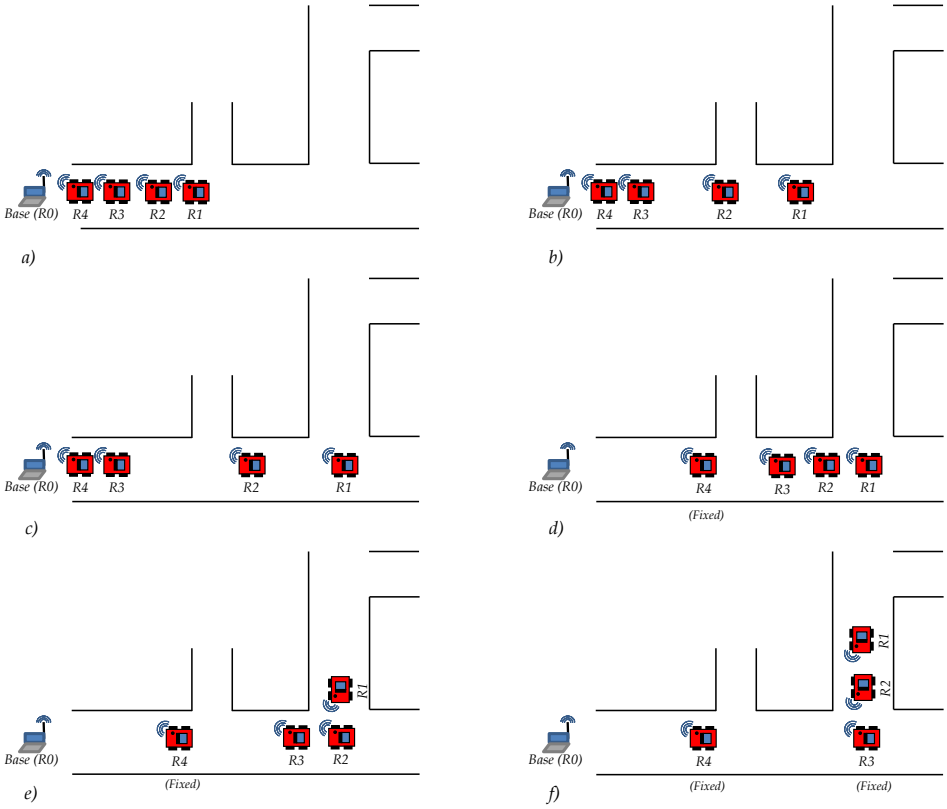


Figure 6.8.: An example of the steps of the algorithm. At the beginning all the robots are placed close to the base station (a). The leader robot starts moving forward looking for the first topological feature specified in the plan of the mission (the second left gallery in this example). At the same time, it is localized at the base station and the laser readings and position provided through the network are used to build simultaneously the map of the environment. When the link quality falls below a certain threshold, the follower R_2 is required to move and starts going after the leader in order to act as a relay in order to provide network connectivity (b). When the leader robot approaches the feature it is looking for, it is stopped and all the slaves are requested to reach the leader (c). During this phase, if the link quality between the base station R_0 and the last follower R_4 falls below the cited threshold, the latter is stopped in its current position, becoming a fixed relay (d). Once the regrouping is complete, the leader enters the feature while, at the same time R_2 takes its place followed by all the still mobile followers (e), allowing the line of sight between both —and thus between each pair or robots — to be guaranteed at every moment. After that, a chain like that of the first shot is set up in the lateral gallery. At this moment the last mobile follower R_3 is fixed in the corner and becomes (another) fixed relay responsible for avoiding the corner and guaranteeing the connection of the robot R_2 with the base station (f).

last_mobile contains the value zero.

The state machine starts with the state *BEGIN* in which the leader robot will move looking for the $n-th$ occurrence of a feature (lines 17, 18). When it finds what it is looking for, it checks if this

Algorithm 3 Deployment Planning

```
1:  state ← BEGIN
2:  plan[] ← read_plan()
3:  last_mobile ← n − 1; id ← 0
4:  set_rope(R_{n−1}, R_0)
5:  for i in 1..n − 2 do
6:    set_rope(R_i + 1, R_i)
7:  end for
8:  while true do
9:    if state == BEGIN then
10:     if found_feature(plan[id]) then
11:       if last_feature(plan[id]) then
12:         break
13:       end if
14:       state ← GROUP_ROBOTS
15:     else
16:       move(R1)
17:       find_feature(plan[id])
18:     end if
19:   else if state = GROUP_ROBOTS then
20:     if robots_grouped() then
21:       compute_feat_pos(&feat_x, &feat_y)
22:       send_xy_goal(R_1, feat_x, feat_y)
23:       state ← ENTER_FEATURE
24:     else
25:       for i in 1..last_mobile − 1 do
26:         send_rel_goal(R_i, R_{i+1})
27:       end for
28:     end if
29:   else if state = ENTER_FEATURE then
30:     if robots_grouped() then
31:       id ← id + 1
32:       fix(last_mobile)
33:       state ← BEGIN
34:     else
35:       for i in 1..last_mobile − 1 do
36:         send_rel_goal(R_i, R_{i+1})
37:       end for
38:     end if
39:   end if
40:   sleep_until_ms(500)
41: end while
```

is the last part of the plan, in which case it exits (line 10, 11). Otherwise, all the robots are grouped behind the leader maintaining the chain formation (lines 23, 24). It is worth remarking that the roping is active at any moment. This means that one or more robots can be stopped during this phase. When the grouping is complete, the position within the feature is computed and the leader robot is sent inside (line 20-22). Then the formation is restored again (lines 35, 36). When all the followers are at their final position, the last mobile node is fixed, and the algorithm restarts after moving the pointer *id* to the next feature (lines 30-32).

The state machine will repeat the same behaviour for each of the tasks included in the plan.

6.5. Experiments

The evaluation of the system has been carried out both by means of simulations and field experiments into two. The first is a maze-like underground mine representing a complex scenario for autonomous deployment. The second simpler scenario is a railway tunnel comprising a long corridor and lateral galleries to experiment with a real robot team deployment, where it is possible to evaluate the whole system working in practice and especially the communication issues.

In the mine scenario, simulations have been carried out to test the robots deployment algorithm, and also to test the recognition of semantic-topological features technique. To do this, metrics to evaluate the rate of correctly recognized features have been established and reported. As the high level plan for the leader and its execution is based on these features being natural landmarks in the environment, it is essential that the recognition algorithm works correctly.

In the tunnel scenario a simulation and two real experiments have been carried out. The simula-

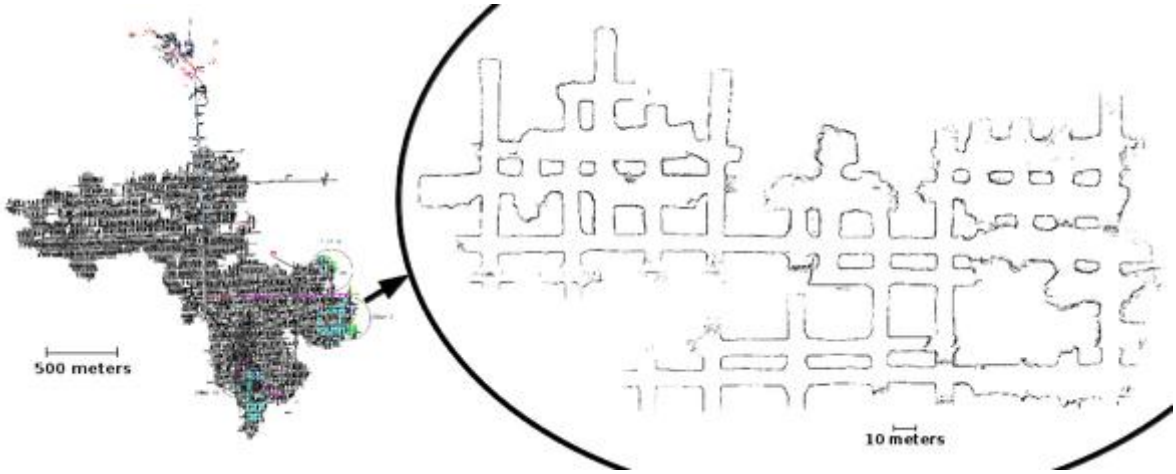


Figure 6.9.: Global map of Santa Marta mine and partial map where simulation experiment has been performed. Courtesy of Minera Santa Marta S.A.

tion is oriented to evaluate the capacity of the system to react against unexpected events, consequently replanning the mission. In the field experiments the system as a whole and all its various parts have been evaluated. This includes the fulfillment of the mission, the topological and metric localization on the map, the correct construction of the geometric map, and the effectiveness of the navigation. Also, the radio signal among the nodes, inter-arrival delay and delay of the messages, the bandwidth allocation and the teleoperation performance have been evaluated. The communications issues assume special importance in these experiments because they cannot be evaluated by means of simulations for obvious reasons. The whole system was implemented over the ROS platform Quigley et al. (2009).

6.5.1. Simulations

As anticipated, two scenarios were selected for the simulation. In the first —a mine—, we focused on the successful completion of quite a complex mission that involves 5 robots which leader had to reach a specific location under limited connectivity conditions after recognizing several features and discarding many others. The second scenario, represented by a tunnel, was used to demonstrate the robustness of the system against unexpected situations such as a sudden drop in signal quality and the need for on-the-fly mission replanning.

The Santa Marta mine

The scenario in which the first simulation was performed, is the Santa Marta mine (see fig. 6.9). A global map and a zoom of the portion used in the experiment is represented in figure 6.9. The global map is schematic and is maintained by Minera Santa Marta S.A. for exploitation purposes while the part used here was previously generated by manually guiding a robot in a data collection exercise carried out previously in the real mine using a Pioneer P3AT robot.

A five robot team was set up using the Stage simulator in the ROS environment. The radio signal was simulated and computed as a function of the Manhattan distance between each pair of robots.

The scenario includes several features among those described in section 6.3.4 in table 6.6. In the simulation experiment, the high level plan can be summarized as: *"Move to the left at the first ⊣,*

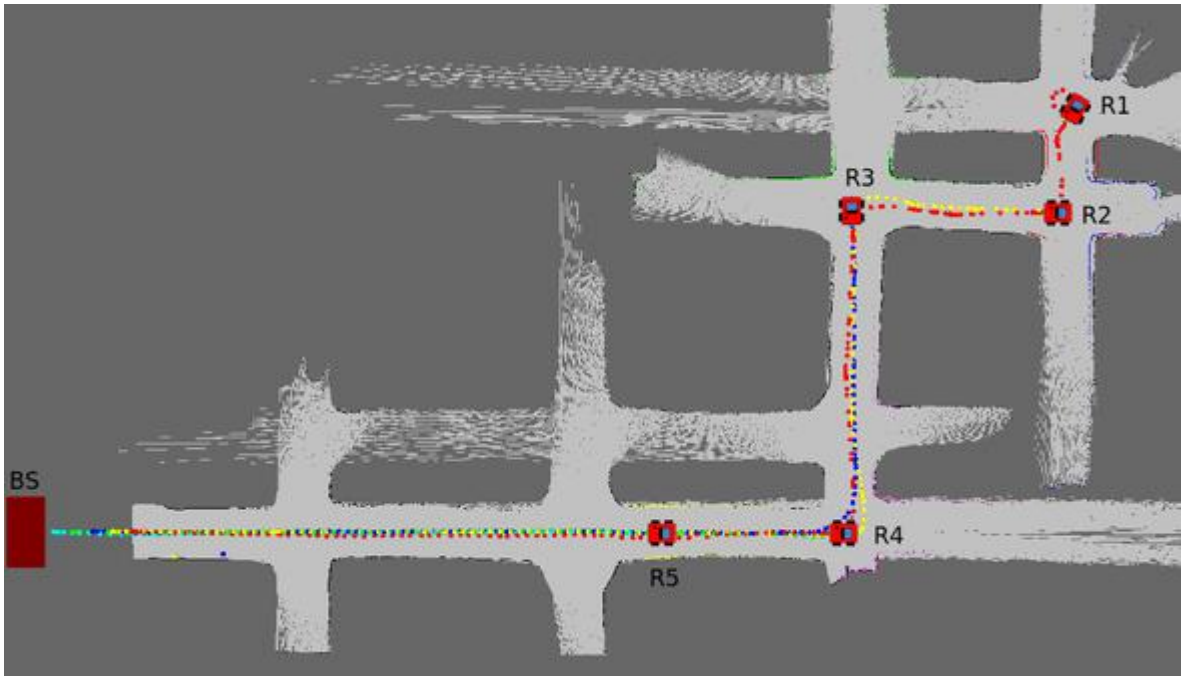


Figure 6.10.: Path followed by the robots in a zone of the mine represented in figure 6.1.a (simulation).

then to the right at the second +, then to the left at the first +, then stop at the first +" and activate teleoperation.

Figure 6.10 shows the path followed by the nodes and the final configuration. Robot R_5 is stopped after the second cross to maintain the communication between R_4 and R_0 (*BS*) while robots R_4, R_3 and R_2 are fixed in the corners to act as relays. Finally, the leader robot is placed in the mission goal position where the teleoperation begins. The complete path is a zig-zag between several left and right galleries, as can be seen in the figure. The final position of the robots permits a line of sight between each pair and guarantees that the distance between them is short enough to enable comfortable communication.

Figure 6.11 represents the posterior probability of the feature detector obtained during the experiment using the non-linear detector. It can be seen that both kinds of features encountered during the exploration (*Cross* and *Left hand*) are identified. We have compared the results of applying both the linear and non-linear detectors explained above and concluded that the non-linear yields better results. After an appropriate filtering, the topological localization based on these semantic features is provided to the state machine that applies the plan according to this information. A video of the experiment can be seen in *

Performance analysis of semantic-topological features recognition technique

In order to evaluate the performance of the semantic-topological features recognition technique we have simulated a set of long enough tours through the mapped area of the mine. The distance traveled was 4.5 km approximately and the number of features involved was 346. Due to the nature of the

*https://youtu.be/ZjLNDHzji4Q

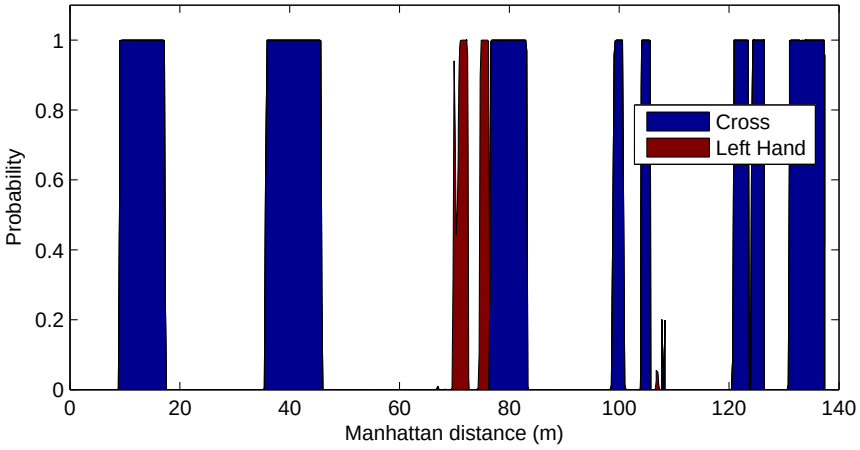


Figure 6.11.: Feature detector probability. In blue $+$, in red $\dashv$, the first to be reached, the only ones appearing in the mission of figure 6.1.

environment we can identify different areas in which the features exhibit different levels of deformation depending upon the current state of the mine exploitation: it is possible to clearly distinguish between the areas that have been already exploited and mine faces that are being currently exploited. Figure 6.12 shows three areas with different levels of exploitation: area *A* corresponds to galleries in which the topological features exhibit little distortions, while areas *B* and *C* correspond to mine faces in which much of the features are heavily distorted. In order to reflect these differences, we have evaluated the feature recognizer performance within areas *A* and *B* separately.

	Area A		Area B	
Feature	Linear	Non-linear	Linear	Non-linear
Diaphanous	100	100	50	11.1
Cross	82.1	100	33.4	56.8
Left/Right Hand	100	100	100	0
Left/Right Turn	77.8	66.7	100	100
End of corridor	100	100	100	100
T-intersection	100	100	55.6	88.9

Table 6.1.: Per class true positive rate obtained with linear and non-linear method on the two areas (%): A (poorly distorted area), B (mine face area).

Table 6.1 shows the success rates obtained in the experiment. They vary from the high success rates that correspond to the poorly distorted features in area *A* to the worse rates obtained within the *B* mine face area. Furthermore, in order to reflect the improvements related to the use of the non-linear classifier, the table shows separate columns for linear and non-linear classification schemes. In general, the non-linear classifier gives better results than the linear one. In section 6.6 we present some lessons learned on this topic. Finally, it is worth remarking that our technique do not require hard computations: each step of our feature recognition algorithm requires less than 10ms for both linear and non-linear versions on a Intel Core i5 3470 processor (3.2 GHz).

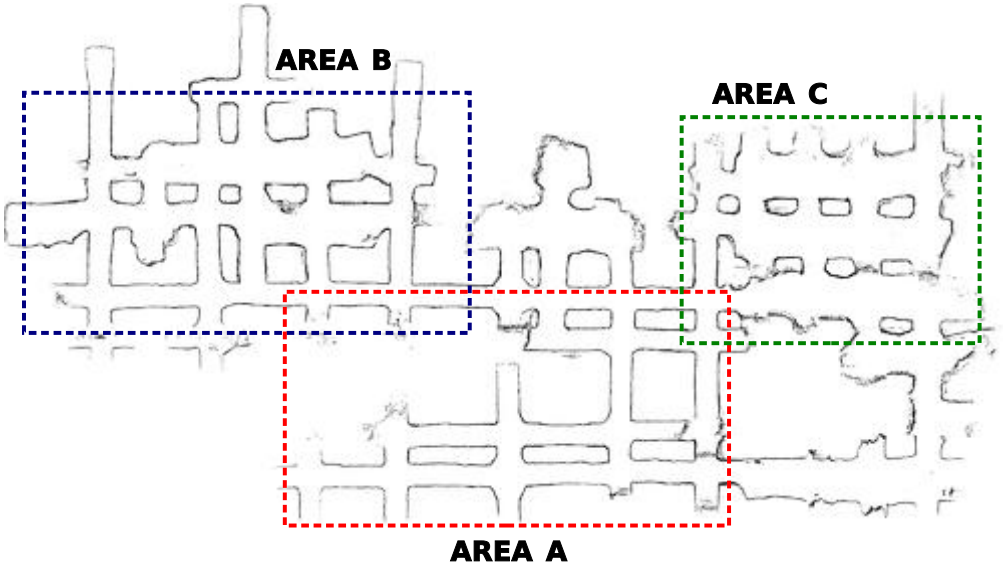
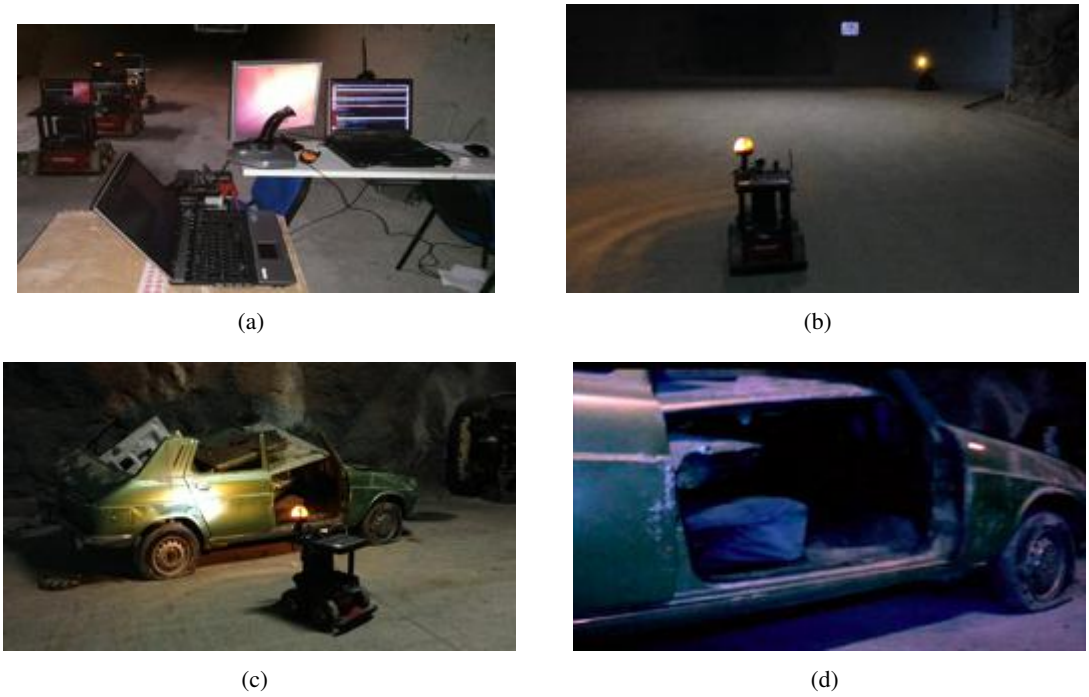


Figure 6.12.: Simulation environment divided into three areas according to level of exploitation.

Figure 6.13.: Snapshot of the robots navigating the Somport Tunnel

6.5.2. Field experiments

After completing the simulations, we carried out a set of field experiments in the Somport Tunnel (see fig. 6.13). This is an old railway tunnel, 7.7 Km long, that is representative of long straight



(a) (b)

(c) (d)

Figure 6.14.: Screenshots from the real experiment. We can see the robot formation and the base station at the starting point (a), how the leader robot approaches the lateral gallery while the follower is fixed as relay (b), the leader robot exploring (teleoperation) the zone of interest (c) and a snapshot received at base station (d).

tunnels typical in transport or mining environments. It connects Spain with France through the Central Pyrenees. It has a horseshoe shape cross section, 6 meters high and 5.5 m wide and 17 lateral galleries, each more than 100 meters in length and, of the same height as the tunnel. The objective was to evaluate the results in a real-world scenario using mobile robots and actual wireless communication.

We used three Pioneer P3AT and one Pioneer P3DX mobile robots controlled by Dell D630 laptops. Moreover at the base station we used a Getac B3000 Rugged laptop with a core *i*7 processor. All the nodes were equipped with Alfa AWUS051NH wireless cards in which the operational frequency was fixed at 2.412Mhz and the data rate at 12Mbps.

The objective of the experiments was the completion of a mission in a scenario that has to be explored by the leader whilst it builds a map in which it is self-localized. In the first experiment, the leader robot has to reach a specific gallery topologically specified in the mission plan, while maintaining communication with the base station at all times. The other robots have to follow the leader, serving as relay communication nodes when needed. Once the destination is reached, the plan specifies that the human operator has to explore the area teleoperating the robot by means of a joystick while receiving video images through the network.

Network profiling

In the first real-world experiment the mission consisted of arriving at a lateral gallery in the tunnel, reaching the end of the gallery, and then teleoperating the robot to inspect a group of cars supposedly involved in a road accident in order to visually analyse the scene from the base station. The base

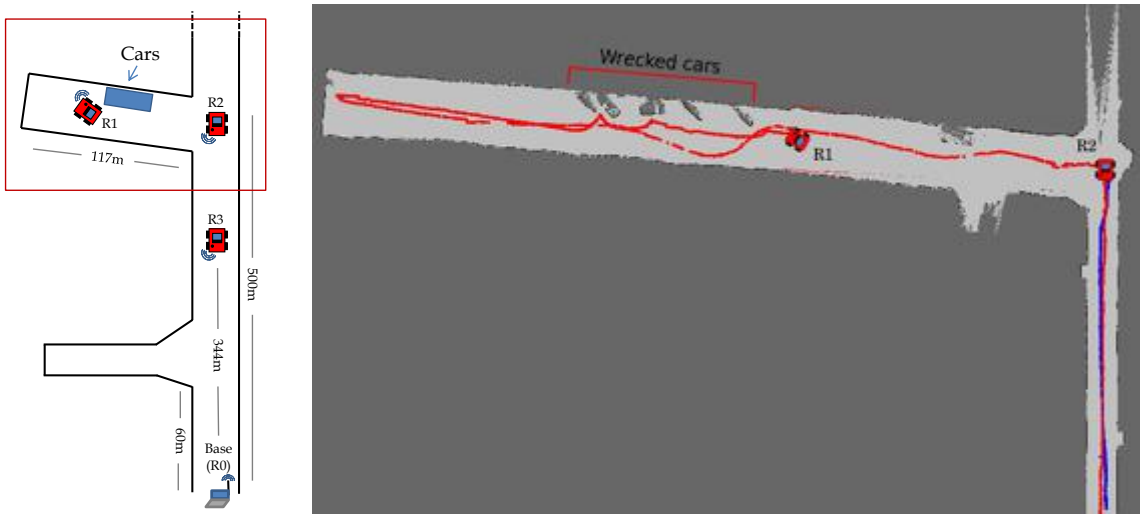


Figure 6.15.: Final configuration of the robots in the field experiment (left) and detail of the zone identified by the red rectangle (right).

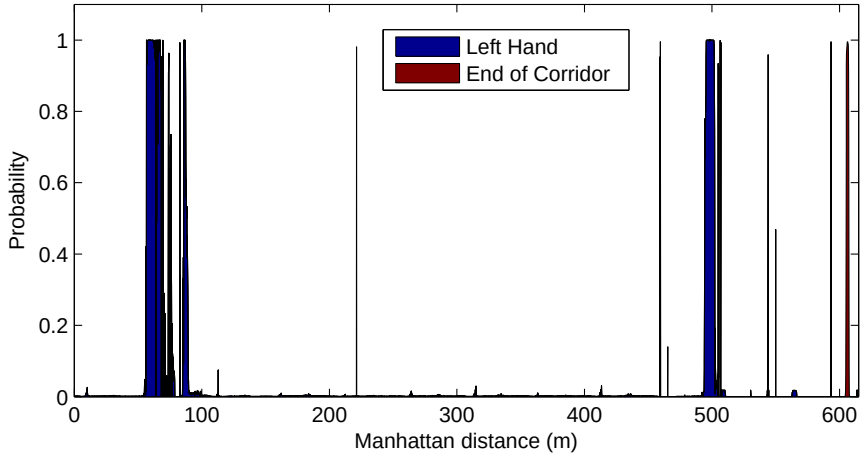


Figure 6.16.: Feature detector probability. In blue, $\dashv$ features detected; in red, $\sqcap$ feature at the end of the corridor. Sporadic potholes detected on the floor appearing in the figure are filtered by the feature detector.

station was established at about 500 meters from the goal gallery. For this mission the topological-level plan was: *"Enter the left gallery at the second $\dashv$, then reach $\sqcap$ and start the teleoperation."*

In the gallery the robots first navigated autonomously. When the leader reached the final goal of the mission, the system changed to teleoperation mode in which the robot was managed by the human supervisor at the base station. Once the zone close to the wrecked cars was reached, the accident zone was explored taking advantage of the front camera of R_1 (see figure 6.14) and the laser feedback. The other robots remained still at their locations in such a way that the leader was continuously connected to the base station. The operator reported that the refresh rate of the laser and video feedbacks was adequate making the teleoperation comfortable and secure. No robot crashes were reported thanks to the RNM module.

Figure 6.17.: Complete map of the environment obtained at the base station.

Figure 6.15 shows the final configuration of the robots and a detail of the final part of the experiment when the leader robot R_1 entered the gallery by means of a turn manoeuvre, once it had identified this intermediate objective of the plan. Robot R_2 acted as a relay node maintaining the connectivity with the base station through node R_3 that had previously been stopped in the tunnel at approximately 344 meters from the base station.

Figure 6.16 shows the posterior probability computed by the feature detector during the experiment, starting at the base station and moving towards the second gallery. The detector clearly identifies the two *left hand* galleries encountered during the exploration at positions $x = 60m$ and $x = 500m$ and the *end of corridor* at position $x = 605m$. The sporadic detection of *end of corridor* features along the path is due to the rotation of the robot on the pitch axis due to the presence of potholes in the floor, instantaneously identified as such because they exhibit a similar laser structure. However, these data are filtered before reaching the state machine. Notice that the feature detector recognizes clearly (high probability) different kinds of galleries (not polygonal, intersection angles $\neq 90^o$, with chamfers, etc.), learned from a set of general artificial scenarios but successfully applied to real environments.

Figure 6.17 shows the complete map of the field experiment scenario (approximate measures $500m \times 120m$) obtained at the base station in real-time. Figure 6.18 shows the map obtained by running the SLAM algorithm offline using the data logged into the leader on-board computer, and the map built in real-time at the base station using the data received through the network. Both maps are very similar and only small differences can be observed between them. This demonstrates that the delay introduced by the network or the sporadic loss of data that may have occurred have not influenced the results of this task. It is worth remarking that the maps shown (those in figures 6.10 and 6.15, specifically) have also been built at the base station side. A video of the experiment can be seen in [†]

6.6. Lessons learned

Setting up such a complex system entails several challenges, as described above. While most potential problems can be foreseen and solutions found with greater or lesser success, other aspects are much more difficult to anticipate and must be evaluated and solved after they occur. In this section we analyse some of the issues we encountered during the development of the system and the solutions we have implemented to overcome them.

[†] https://youtu.be/BpxHXTZjKF4

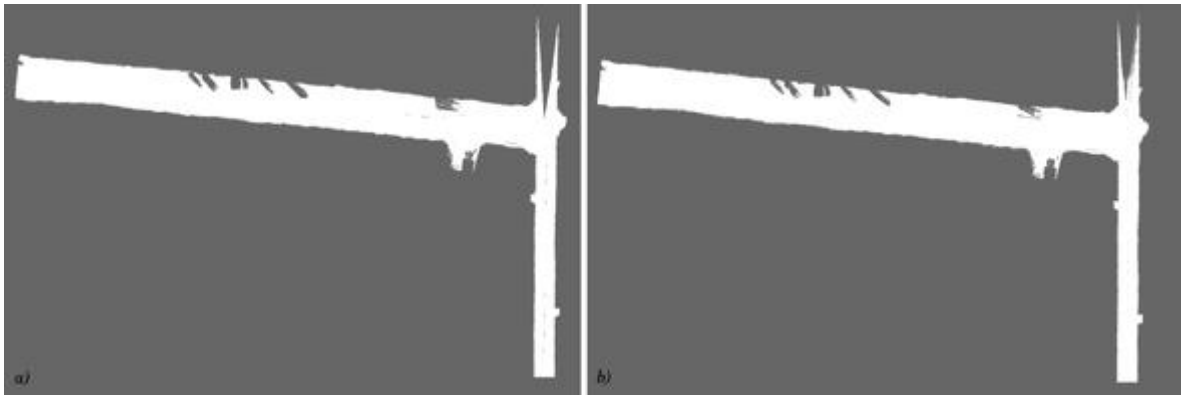


Figure 6.18.: Maps obtained offline (a) and online (b).

6.6.1. Localization

One of the most important problems is the localization of a robot team in this kind of environment due to its huge size and the fact that often the ground is covered with stones or potholes, which disturb the navigation and so the localization. Our initial view was that an exact localization was not necessary since the robots would be guided in terms of semantic features. However, the localization of the robots relative to each other is important given that this is used for relative movement and positioning. Thus, instead of localizing all the robots globally, we decided to use the leader robot to build a map and localize the others over the same map. This means that, even if the global localization may be imprecise, the relative localization is sufficiently precise.

After a set of experiments using different kinds of SLAM algorithms among those already implemented and available in the ROS platform, we selected the *GMapping* algorithm as the most robust. On the other hand, we selected a particle filter to localize the followers. However, at first the algorithm did not work well and the robots often got completely lost, especially when a pothole or a pebble caused the robot to abruptly changes its orientation. To solve this problem, on the one hand and we considerably relaxed the confidence of the particle filter as if the odometry was much worse than it actually is. Given that the robots can move for long periods without having reliable features to which to refer (e.g. a long stretch with smooth walls), the MCL filter tends to underestimate the movement. When a reliable feature appears (e.g. a cross), it is no longer able to fit the laser reading and the map due the fact that the group of particles, even if scattered, is too far from the real position. On the other hand, we changed the settings to make the filter believe that the robots are holonomic. In this way, the particle filter also takes into account the lateral displacements that sometimes occur in this kind of terrain, and is able to recover a correct localization even when unusual movements occur.

6.6.2. Navigation

The navigation in this kind of environment is not straightforward, either. In some scenarios, the leader robot has to cover long distances relying only on basic data using a laser and odometry at the highest possible speed and without crashing against the walls. In other words, it must go straight centred in the environment to avoid crashing. Our initial idea was to use the method cited in section 6.3.3 which consists of analysing the laser through the Hough transform to identify the walls and provide a local goal for the robot at a fixed distance from one of the walls. Even though this method works

satisfactorily in most environments, we came to realise that in environments in which the lateral walls are not sufficiently smooth or straight, this method would fail given that the Hough transform is not able to return a reliable identification. We decided to implement the additional solution described, where the laser readings are analyzed to identify maxima that are then used as robot goals. The joint use of these two methods guarantees a safe and effective navigation of the leader robot with a small computation cost.

Another issue regarding navigation in those scenarios is the existence of potholes and a irregular terrain. Also errors in localization are increased. Navigation techniques based in occupancy or traversability maps can be useful in many cases to have a more robust performance.

6.6.3. Semantic feature recognition

Several issues appeared during the experiments, all of them related to the nature of the environment, that strongly influence the recognition performance shown in table 6.1, requiring detailed analysis for the following situations (see figure 6.19): overlapped features, ambiguous features, heavily distorted features.

(a) (b) (c)

Figure 6.19.: Different features found in the environment: (a) overlapped feature, (b) ambiguous feature, (c) heavily distorted feature.

There are situations in which two identical features (e.g. two + features) are close to each other enough to appear in the same laser scan (see fig. 6.19(a)). In such cases, the recognition system is unable to indentify the short corridor between the two of them, returning the + feature without any transition. This behaviour, enhanced by the filtering procedure (see section 6.3.4), causes that the two features are considered as an unique one. In order to avoid this issue, we suggest for further improvements the use of the robot odometry (or another local method for localization) to reset the recognition system once the robot leaves the first feature. On the other hand, our environment includes features that could be classified indistinguishably into different classes. These features usually include one or more truncated (or partially occluded) branches (see fig. 6.19(b)). Regardless of whether feature has been detected, the feature detector always reflects correctly the opened branches, allowing us to navigate properly through the feature. Finally, there are some areas of the environment (e.g. the area C in figure 6.12) in which the features are too heavily distorted with respect to the training ones (see fig. 6.19(c)), jeopardizing their recognizability. Moreover, it could make sense to consider the referred area as a big column hall, rather than a maze-like scenario. All these detected problems

suggest that there is room for improvements in the learning and recognition techniques in challenging scenarios as the found in some areas of the environments used in the experiments.

6.6.4. Communication

We realized that the bandwidth required by this scheme can be the bottleneck for its scalability. To reduce the demand, we set up a technique to avoid the transmission of unnecessary and redundant data reducing, for example, the frequency of the laser flows when the robots are still.

A final consideration is that for reaching longer deploying distances, deployment planning and navigation techniques as those presented in Rizzo et al. (2013) should be used.

6.7. Discussion

We have presented a complete framework for safety, security, intervention or rescue tasks in structured environments such as tunnels and mines where there are usually no communication infrastructures. The framework is based on a multi-robot team and has been designed for use in situations where the access of human intervention teams can be dangerous or harmful (in case of fire, chemical or radioactivity contamination, etc.). The goal is to deploy the team in an autonomous manner, providing the system with a mission plan specified in terms of topological-semantic features (e.g. third left gallery, then fourth cross, etc.). The deployment is supervised from a base station where a human operator receives laser and video feedback from a leader robot and has the possibility of teleoperating it through a radio frequency link established over a multi-hop path. A map of the environment is also built at the base station for possible subsequent human intervention. The platform has been implemented and tested in simulations and in field experiments. The results show that the solution is feasible and offers satisfactory results in terms of autonomy, teleoperation possibilities, quality of map building and quality —in terms of timing— of the data received at the base station.

In this work we propose improvements in: i) localization based on semantic-topological features; ii) navigation planning based on the natural landmarks learned; iii) coordinated robot deployment and iv) real-time and multi-hop communication network. A complete system integrating all those techniques jointly to others well tested algorithms has been developed.

This research work has also shown that in robotics applications in real and large environments all the aspects considered have to be fully integrated. We have shown that topological localization provides greater robustness in navigation and localization while exploring an unknown scenario, given that the use of semantic features to topologically localize robots allows missions to be described without the need for detailed metric maps. Safe and robust navigation is also mandatory, especially when the possibility of remote human actuation is limited.

Chapter 7

Conclusions

This chapter presents a summary of the most relevant conclusions obtained during this thesis. As a general conclusion, it is clear that the insertion of semantic information into the point maps created by SLAM algorithms makes it possible for a robot to be able to understand the environment in which it moves, to significantly improve the way it makes decisions and navigates and even to interact with it.

Firstly, a proof of concept of semantic SLAM has been carried out. The chapter 2 presents a system consisting of a monocular EKF SLAM and a basic object detector. This synergy, which provides a partially annotated local map and the current position of the robot on this map, is of great value for the tasks performed by service robots. One of the main advantages of this method is that the models used for recognition can be generated with a different camera from that used to perform visual SLAM, which provides a greater degree of flexibility to the system because the task can be defined in terms of the objects the robot has to interact with, in contrast with plans based in terms of spatial coordinates. The main weakness of this proposal was the poor scaling both in the geometrical map size and in the object database size. The limitations in the geometrical map comes from using EKF SLAM, that was superseded by the keyframe and bundle adjustment based methods. These issues were therefore addressed separately in the following chapters.

In chapter 3 we focused our efforts on designing and implementing a visual SLAM that exploits the benefit provided by the methods based on keyframe selection and non-linear bundle adjustment optimization. The main contribution was to outsource to the cloud those processes that require more memory or CPU. In our case, we have moved the mapping and optimization process to the cloud servers, achieving satisfactory results without loss of performance and maintaining strict real-time requirements. In addition, having the maps stored on a common server makes it possible for other robots to reuse these maps. The experimental validation confirms that using commodity wi-fi channel, the performance of bundle adjustment VSLAM methods can be delivered with limited computing resources in the on-board computer.

The next step was to use this semantic information inserted in the maps to improve the classic navigation of the robots in metric maps. Experimental validation of a pioneer semantic mapping system using cloud computing has been provided. For robot planning we use the agnostic to the robot *action recipes*. In the experiments, it is shown, how semantic content of the maps are crucial for grounding the generic action recipe to generate a specific plan for the actual robot in the current scene that efficiently searches for an object in the workspace. The object detection scalability has been also improved compared to the one presented in chapter 2 by adding a Bag of Words based

implementation, which boosts the object database size from a few tens to a few hundred or even thousands of objects. In addition, an ontology has been added to represent the knowledge stored in these maps. Thanks to this combination, knowledge-based reasoning on the map is possible. The proposed system demonstrates how a robot is able to generate semantically significant environment maps and reasoning about them in conjunction with a formally pre-established knowledge. Regarding the impact on improving navigation and searching for objects, the system presented is able to reduce by 70% the number of potential positions that the robots have to visit to cover an environment.

In chapter 5 the issue of localization and navigation in environments where there are a large number of people moving around is addresed. Existing SLAM techniques offer robust performance in rigid scenes. The observation of populated environments implies non persistent changes of scene and regions of scene with non rigid movements. For this reason, a system able to operate robustly in populated environments was developed by inserting a person detector into the SLAM algorithm to identify and eliminate them from the global map to improve the static map and the robot localization. This process not only benefits the scene description, but also the performance of the SLAM algorithm when the robot is in populated environments. The method presented has been validated with two visual SLAM systems, obtaining a similar performance improvement in both. The Absolute Trajectory Error (ATE) selected for measuring the performance of the mapping system is reduced by more than half if human activity masking is applied to the images. Regarding the 3D reconstruction of the environment, in the validation experiments performed, the number of points in the scene is reduced in a 30% thanks to the removal of people from the scene. Therefore, we can conclude that it is agnostic to the SLAM method, and therefore can improve any SLAM system. In addition, the implementation carried out offers the possibility of changing the detector used, therefore it is also agnostic to the recognition method used. This gives great flexibility to the system presented.

Finally, in chapter 6 we show how the main contribution of the tesis, distributed maps containing semantic information, are mandatory for autonomous operation, with a example in a multirobot system operating in tunnels and mines. The architecture developed integrates all the necessary functionalities to address the challenges of these environments, such as: localization based on topological features of the environment, navigation and planning based on natural landmarks of the environment, coordinated deployment of robots based on a high-level plan, and map building. This system has been validated with satisfactory results in real confined environments, which present many challenges. Having semantic information about the environment has been crucial to detail the missions of robots with a high-level language.

7.1. Future work

We can conclude that the semantic information significantly improves robots autonomy. The future work should be oriented boost the semantic content of the maps.

On the one hand, it would be interesting to recognize generic categories rather than objects in terms of the detection and insertion of semantic information. In addition, context-based recognition or image segmentation would increase the amount of information we could insert into the maps. Another aspect to take into account would be to determine which information captured from the environment is relevant for insertion in the map.

Secondly, the role that cloud computing and cloud robotics can bring. In our case, this future work would go along the lines of exploiting the cloud computing capability in order to be able to execute more complex algorithms that cannot be performed in a robot. For example, recognition in images would be improved if we applied machine learning techniques. The computation required for

these techniques could be outsourced to the cloud and a robot with less computing power capabilities could use these techniques.

Finally, the possibility of detecting people and inserting them in the map, presented in chapter 5, opens up a line of research inspired by the following questions: What benefits does it bring to know the location of people on a map? Can a robot improve its navigation in an environment knowing the location of people on it? Is it possible to plan a better trajectory taking into account the flow of movement of people? Answering these questions poses new challenges for future work related to movement of robot taking into consideration not only the semantic of the objects and places in the scene, but also the semantics of the people around the robot.

Chapter 8

Conclusiones

En este capítulo se presenta un resumen de las conclusiones mas relevantes obtenidas a lo largo de esta tesis. Como conclusión general, queda patente que la inserción de información semántica dentro de los mapa de puntos creados por los algoritmos de SLAM posibilita que un robot sea capaz de entender el entorno por el que se mueve, pueda mejorar de una forma significativa la manera que tiene de tomar decisiones y navegar e incluso le dé la capacidad de interactuar con él.

En primer lugar, se ha realizado una prueba de concepto de SLAM semántico. El capítulo 2 presenta un sistema compuesto por un EKF SLAM monocular y un detector básico de objetos. Esta sinergia que proporciona un mapa local parcialmente anotado y la posición actual del robot en él, es de gran valor para las tareas realizadas por robots de servicio. Una de las principales ventajas de este método reside en que los modelos utilizados para el reconocimiento pueden ser generados con una cámara diferente a la utilizada para realizar el SLAM visual, lo que porporciona un mayor grado de flexibilidad al sistema, ya que la tarea puede definirse en términos de objetos con los que el robot debe interactuar, en contraste con los planes basados en coordenadas espaciales. La principal debilidad de esta propuesta fue la escasa escala tanto en el tamaño del mapa geométrico como en el tamaño de la base de datos de objetos. Las limitaciones en el mapa geométrico provienen del uso de EKF SLAM, que fue reemplazado por los métodos basados en ”keyframes” y ”bundle ajustment”. Por ello en los siguientes capítulos se abordaron por separado estos aspectos.

En el capítulo 3 hemos centrado nuestros esfuerzos en diseñar e implementar un algoritmo de SLAM que aprovecha el beneficio proporcionado por los métodos basados en ”keyframes” y la optimización usando ”bundle ajustment”.

La principal contribución ha sido externalizar en la nube aquellos procesos que demandan mas memoria o CPU. En nuestro caso, hemos movido a los servidores en la nube el proceso de construcción de mapas y la optimización, obteniedo unos resultados satisfactorios sin pérdida de rendimiento y manteniendo los estrictos requisitos de tiempo real. Además, tener almacenados los mapas en un servidor común posibilita la reutilización de estos mapas por parte de otros robots. La validación experimental confirma que utilizando un canal wi-fi estandar, el rendimiento de estos métodos de VSLAM permite que el sistema pueda funcionar con los recursos informáticos limitados que tiene el robot.

El siguiente paso realizado ha consistido en utilizar esa información semántica insertada en los mapas para mejorar la navegación clásica de los robots en mapas métricos. Para la planificación de robots utilizamos *recetas de acción* agnósticas al robot. En los experimentos, se muestra cómo el contenido semántico de los mapas es crucial para proporcionar información a la receta de acción

genérica y proporcionar una plan específico para el robot en la escena actual. De este modo realiza una búsqueda eficiente de un objeto en un entorno. La escalabilidad del método de detección de objetos ha mejorado también con respecto al presentado en el capítulo 2 añadiendo una implementación basada en bolsa de palabras. Además, se ha añadido una ontología para representar el conocimiento almacenado en esos mapas. Gracias a esta combinación, el razonamiento basado en el conocimiento sobre el mapa es posible. El sistema propuesto demuestra como un robot es capaz de generar mapas de entorno semánticamente significativos y razonar sobre ellos en conjunción con un conocimiento formalmente preestablecido. En cuanto al impacto en la mejora de la navegación y la búsqueda de objetos, el sistema presentado es capaz de reducir en un 70% la cantidad de posiciones potenciales que los robots tienen que visitar para cubrir un entorno.

En el capítulo 3 se planteó la problemática que tiene un robot a la hora de localizarse y navegar en entornos donde hay un gran número de personas moviendose por la escena. Las técnicas de SLAM existentes ofrecen un rendimiento robusto en escenas casi siempre rígidas. Por el contrario la observación de entornos poblados implica cambios de escena no persistentes y regiones de escena con movimientos no rígidos. Por ello se desarrolló un sistema capaz de funcionar con robustez en entornos poblados mediante la insercción dentro del algoritmo de SLAM de un detector de personas que ayuda a identificarlas y eliminarlas del mapa global de la escena, mejorando los mapas estaticos y la localización del robot. Este proceso no sólo beneficia a la descripción de la escena, sino también al rendimiento del algoritmo de SLAM cuando el robot se encuentra en entornos poblados. El método presentado ha sido validado con dos sistemas de SLAM visual, obteniendo una ganancia de rendimiento similar en ambos. El Error de Trayectoria Absoluta (ATE) seleccionado para medir el rendimiento del sistema de construcción de mapas se reduce en más de la mitad si se aplica el enmascaramiento de la actividad humana a las imágenes. En cuanto a la reconstrucción 3D del entorno, en la experimentación relizada, el número de puntos en la escena se reduce en un 30% gracias a la eliminación de las personas presentes en la escena.

Por lo tanto, podemos concluir que es agnóstico al método SLAM, y por lo tanto puede mejorar cualquier sistema SLAM. Además la implementación realizada ofrece la posibilidad de cambiar el detector utilizado, por lo tanto también es agnóstico al método de reconocimiento. Este hecho dota de una gran flexibilidad al sistema presentado.

Por último en el capítulo 6 mostramos cómo la contribución principal de la tesis (mapas distribuidos que contienen información semántica), es crucial para el funcionamiento autónomo de un robot, como por ejemplo en un sistema multirobot operando en túneles y minas.

Se ha desarrollado una arquitectura que integra todas las funcionalidades necesarias para abordar los retos encontrados en esos entornos, como por ejemplo: localización basada en características topológicas del entorno, navegación y planificación basada en puntos de referencia naturales del propio entorno, despliegue coordinado de robots atendiento a un plan de alto nivel, construcción de mapas. Este sistema ha sido validado exhaustivamente con resultados satisfactorios en entornos reales confinados, los cuales presentan muchos desafios. Disponer de información semántica del entorno ha posibilitado que la misión a realizar por los robots se pueda detallar en un lenguaje de alto nivel a modo de instrucciones.

8.1. Trabajo futuro

Podemos concluir que la información semántica mejora significativamente la autonomía de los robots. El trabajo futuro debe orientarse a potenciar el contenido semántico de los mapas.

Por un lado, en cuanto a la detección e inserción de información semántica, sería interesante

poder reconocer categorías genéricas en lugar de objetos. Además un reconocimiento basado en contextos o una segmentación de la imagen aumentaría la cantidad de información que podriamos insertar en los mapas. Otro aspecto a tener en cuenta sería el determinar que información capturada del entorno es relevante para ser insertada en el mapa.

En segundo lugar, el papel que la computación y la robótica en la nube pueden aportar. En nuestro caso, este trabajo futuro iría en la línea de explotar la capacidad de computo de la nube para poder ejecutar algoritmos mas complejos que en un robot no podrían realizarse a la frecuencia adecuada. Por ejemplo, el reconocimiento en las imagenes se vería mejorado si aplicamos técnicas de machine learning. El cómputo que requieren estas técnicas podría externalizarse en la nube y un robot con poca capacidad de cálculo podría utilizarlas.

Finalmente, la posibilidad de detectar personas e insertarlas en el mapa presentado en el capítulo 5 abré una linea de investigación planteando las siguientes cuestiones: ¿Qué beneficios aporta saber la ubicación de las personas en un mapa? ¿Puede un robot mejorar su navegación en un entorno conociendo la ubicación de las personas en el mismo?, ¿Es posible planificar una mejor trayectoria teniendo en cuenta el flujo de movimiento de las personas?. Estas preguntas plantean un trabajo futuro en relación a la planificación de movimiento de un robot teniendo en cuenta no sólo la semántica de los objetos y lugares en la escena, sino también la semántica de las personas que rodean el robot.

Appendix A

Creating and using RoboEarth object models

A.1. Introduction

This work introduces a way to build up and use an extensive sensor-independent object model database. In a first step, a cost-effective and computationally cheap way to create colored point cloud models from common household objects by using a Microsoft Kinect camera Microsoft (2011) is presented. Those object models are stored in a world-wide accessible, distributed database called RoboEarth RoboEarth Consortium (2011), Waibel et al. (2010). Finally, the models are used for recognizing the corresponding objects with any kind of camera. In the presented implementation the demonstration was done with both a Kinect and common RGB cameras. The implementation is available as a set of ROS Quigley et al. (2009) packages.

A.2. Related work

There are multiple instances of 3D object databases available on the Internet today. Popular examples include Google 3D Warehouse Google (2011), the KIT object model database Karlsruhe Institute of Technology (KIT) (2011) or Willow Garage's household objects SQL database Willow Garage (2011). A common property of these databases is that the object models are stored as triangular meshes. They are mostly of high quality, but object creation requires either a lot of manual work or expensive scanning equipment.

In contrast, the focus of the work presented in this video is on providing a simple and cost-effective way to create object models for object recognition and pose estimation. Instead of triangular meshes, object models are stored as 3D colored point clouds.

A.3. Recording arbitrary objects

For the object recording process the Kinect camera is used in conjunction with a marker pattern. Before starting the object recording process, the target object is placed in the center of the predefined

Figure A.1.: Object recording setup (left) and merged point cloud (right)

marker pattern on a table (see figure A.1 left).Subsequently, the marker pattern is rotated slowly by the user so that the Kinect camera records the object from different views.

The ARToolkit library Kato and Billinghurst (1999) is used to extract the approximate positions of the markers in the camera's RGB image. To further improve the precision, two lines are selected that cross in the center of the marker. Using the depth information from the Kinect, we sample at least six 3D points from each line and apply a least-square fitting approach to estimate the line parameters and the marker's position. The intersection point of these two lines gives a better estimation for the respective marker's center point. In case the lines do not meet, the marker's center point is defined as the point where the distance of both lines is minimal, as long as this distance is below a given threshold.

To discard implausible marker positions, we compare the detected marker positions with the a-priori known relative distances between the markers on the marker template. If at least three of the detected marker center points are classified as plausible in this manner, a coordinate system is established with its origin in the center of the marker pattern by applying the Gram-Schmidt process (e.g. in Arfken et al. (2011)).

Finally, the different recordings can be transformed into the marker pattern center coordinate system and merged into the final object model (see figure A.1 right for an example). For further details on this we refer to Koch et al. (2011).

A.4. Database

In the next step, the data created in the recording process is compressed and uploaded to the RoboEarth database, a Hadoop-based distributed database accessible over the Internet. The user may annotate the recordings with an object class, name and a free-form, human-readable description. Also, a simple OWL description is generated and uploaded to make the model usable in future knowledge processing. We use KnowRob Tenorth and Beetz (2009) as knowledge processing framework. The description of the database can be found in Schiessle et al. (2011).

A.5. Object detection and pose estimation

The user may download one or more of the object model files stored in the RoboEarth database and use them for object recognition and object pose estimation. Each object model consists of several recordings from different points of view. For each recording, a 3D point cloud along with the SURF features Bay et al. (2006) associated to some of those points are stored.

There are currently two different algorithms implemented that make use of the objects stored in the database, one for using common RGB cameras and an CONCLUSION In this work we presented an approach to create 3D object models for robotic and vision applications in a fastother one for use with the Kinect. Though the recognition process could be moved onto the RoboEarth servers, they are currently implemented to run on the client side. In principle the models can be used with any kind of camera.

Figure A.2.: Pose estimation using a Kinect camera and the method presented in section IV-B. The arrows indicate the position of the camera. The marker pattern is visible on the left.

A.5.1. RGB camera

When an image is acquired with an RGB camera, SURF features are extracted. Then correspondences are calculated between each view of the object model and the camera image. Using the RANSAC algorithm, at least five correspondences that describe a valid transformation from the model image to the camera image are searched, and the object pose is estimated by solving the Perspective-N-Point problem. This approach is described in detail in Civera et al. (2011).

A.5.2. Kinect camera

The detection method employed for use with a Kinect sensor is similar to the method presented in the previous subsection. In this case, the pose is estimated with a rigid transformation between the model

and the camera point clouds. Additionally, the depth information for the feature points estimated in the camera image is used to compare the distances between given feature points with the distances in the respective object model feature points. This check is used to discard more implausible correspondences. An example for successful pose estimation in a point cloud acquired from a Kinect camera is shown in figure A.2.

A.6. Conclusion

In this work we presented an approach to create 3D object models for robotic and vision applications in a fast and inexpensive way compared to established approaches. By using the RoboEarth system for storing the created object models users have world-wide access to the data and can immediately reuse a model as soon as it was created and uploaded. The approach shows general applicability for different kinds of cameras. In this work this was shown by two example implementations for the recognition process of objects. The quality of the recognition can be verified in the video. Combined with the knowledge saved in the RoboEarth database the objects can also be properly classified.

The complete software in conjunction with the RoboEarth platform is already available for download. For further details please see RoboEarth Consortium (2011).

Appendix B

Real-Time 3D Reconstruction using the Cloud

B.1. Introduction

Cloud computing is an emerging technology in the last years. The ability to use this technology in robotics opens a new line of reseach called Cloud Robotics Goldberg and Kehoe (2013). This approach to robotics allows robots to benefit from the massively computational and storage resources of cloud data centers. In addition, the increment of network bandwidth will reduce the transport delays and hence make possible the computation offloading.

During the last years some robotic frameworks had addressed the cloud computing paradigm. Rapyuta Hunziker et al. (2013) presents an open source platforms-as-a-Service (PaaS) cloud framework for robotics applications that enables to offload heavy computation in the cloud. The DAvinCi project Arumugam et al. (2010) proposes a frameworks based on Hadoop cluster with ROS (Robotic Operating System) that provides the scalability and parallelism advantages of cloud computing.

Regarding the 3D reconstruction field, the incipient development of new low-cost sensors, such as Microsoft Kinect or Asus Xtion, has led to the development of new tools to build 3D models of the environment. These sensors are capable of adding the depth information of a scene together with the RGB image that the camera provides. Thus, the sensor provides complete information of the environment. Several approaches like Kinect Fusion, RGBDSlam, C^2TAM Riazuelo et al. (2014b) have been developed in the last years. The goal of theses applications is to provide a continuous localization of the camera sensor while a 3D map of the environment is built. In particular, C^2TAM provides a distributed framework for cooperative tracking and mapping.

B.2. System overview

The presented work in based on an open source technology developed under the European project Roboearth Waibel et al. (2010). Figure B.1 shows a scheme of the 3D reconstruction method presented. The system is mainly composed by 3 pieces: a tracking client, a map builder server and a 3D visualizer. For the visual slam implementation we have used the C^2TAM framework, developed by the author of this contribution. As we mention before this framework has been developed for a distributed execution an fits very well with the cloud approach. For the execution of a component in a cloud

server we use the Rapyuta framework that enables the execution of a software component in a cloud server. In this work, we use Amazon EC2 Amazon Inc. (2012) servers for computing offloading.

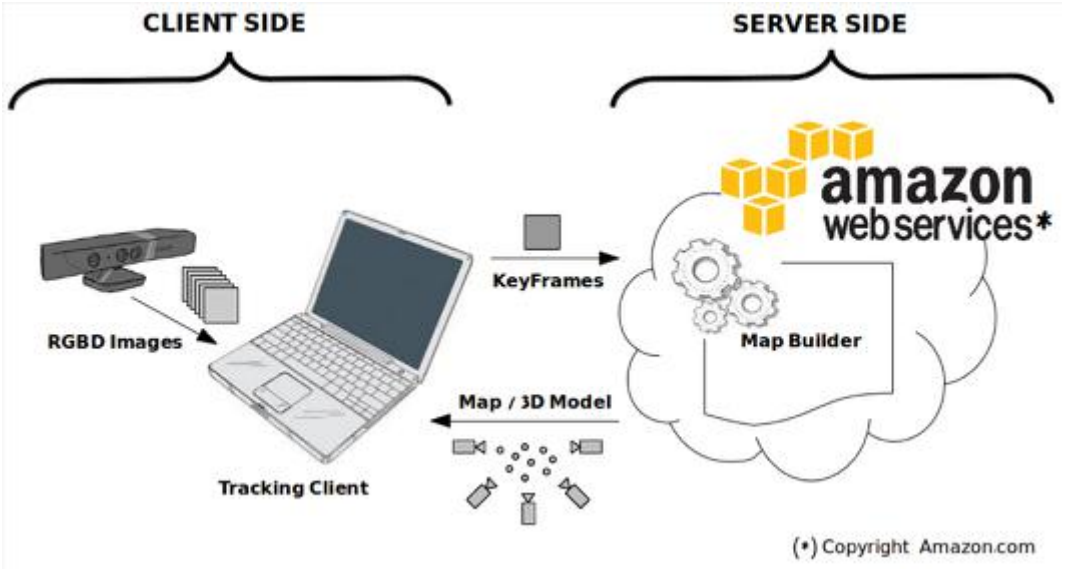


Figure B.1.: System overview of the 3D reconstruction method presented.

B.2.1. Client-side

The tracking component provides a continuously localization of the camera position estimating its position using local descriptors based on salient FAST features on the camera image and the map provided by the server after the optimization in the cloud. Tracking component works at 30 fps and it is placed on the local computer (see figure B.1) where the camera is plugged. The tracking is a light-power process that can be run on a low-power computer, in our case a laptop. It is in charge of select the keyframes of the video stream that will be sent to the server in order to build the map. It is essential to correctly choose the frequency in which these frames are sent to the server because the system must to take into account the network bandwidth and deal with the network delay to avoid a bad performance of the system.

B.2.2. Server-side

The mapping component creates a 3D model of the environment using the data provided by the tracking client (a collection of several keyframes composed by a RGB image and depth information). As we can see on figure B.1, the mapping component runs on an Amazon EC2 server. Rapyuta Cloud Engine provides the execution of this component in a cloud server. The estimation of the 3D map is the most demanding computation of this approach because it requires an optimization process. Due to this process does not require a strong real-time constraints it can be executed in the cloud. 3D model visualizer

The visualizer provides a real-time representation of the 3D model of the environment that is being built and the location of the camera (see figure B.2 for a real experiment visualization). The main advantage of this component is that is able to run both the server side and the client, minimizing data traffic in the network to reduce the bandwidth consumed.

B.3. Experimental results

On this section we present a 3D reconstruction of a real environment using the proposed system. We use a Microsoft Kinect camera that provides RGBD images with a resolution of 640x480 at a frequency of 30Hz, a low-power laptop (AMD Fusion, Asus EeePC 1600.0 MHz) and a Amazon EC2 instance (m1.medium, Intel Xeon). According to the schema proposed on the figure B.1 , the tracking process is allocated on the laptop and the map builder runs in the cloud server. For communication issues we use a standard wireless network. We perform a real time map of a hospital room (see figure B.2 left). The tracking estimates continuously the position of the camera during the experiment and the mapping process builds a map using the keyframes. As a result, figure B.2 right shows the 3D model of this environment generated. A video of the full experiment, and the 3D model generated can be found at *..

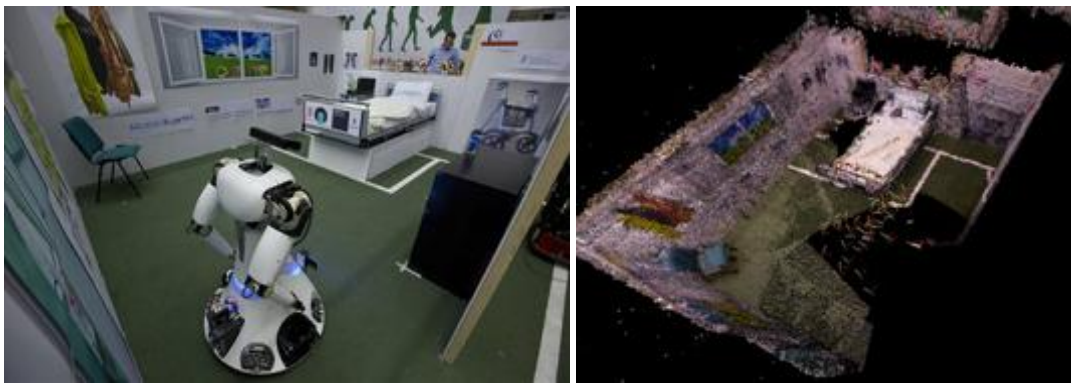


Figure B.2.: Experimental environment (left image) and 3D model of this environment generated (right image).

B.4. Conclusions

This contribution presents a 3D reconstruction of an environment using a low-power computer. We believe that this result opens a new research line in the field of low-cost devices and computation offloading. Using a cloud server for allocating the most expensive tasks, enables the execution 3D reconstruction in real time.

*https://youtu.be/_zaDoBYGr9I

Bibliography

ABAD, P., FRANCO, M., CASTILLON, R., ALONSO, I., CAMBRA, A., SIERRA, J., RIAZUELO, L., MONTANO, L., AND MURILLO, A. C. 2017. Integrating an autonomous robot on a dance and new technologies festival. In *ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Vol. 1. (Cited on page 11.)

ACHTELIK, M. W., WEISS, S., LYNEN, S., CHLI, M., AND SIEGWART, R. 2012. Vision-based MAV Navigation: Implementation Challenges Towards a Usable System in Real-Life Scenarios. In *Workshop on Integration of perception with control and navigation for resource-limited, highly dynamic, autonomous systems, in Robotics: Science and Systems (RSS)*. (Cited on page 29.)

AMAZON INC. 2012. Amazon elastic compute cloud. *Available at* `http://aws.amazon.com/ec2/`. (Cited on pages 60 y 122.)

ANDRILUKA, M., ROTH, S., AND SCHIELE, B. 2009. Pictorial Structures Revisited: People Detection and Articulated Pose Estimation. (Cited on page 75.)

ARFKEN, G., WEBER, H., AND HARRIS, F. 2011. *Mathematical Methods for Physicists: A Comprehensive Guide*. Elsevier Science. (Cited on page 118.)

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., ET AL. 2010. A view of cloud computing. *Communications of the ACM 53,* 4, 50–58. (Cited on page 30.)

ARUMUGAM, R., ENTI, V., BINGBING, L., XIAOJUN, W., BASKARAN, K., KONG, F. F., KUMAR, A., MENG, K. D., AND KIT, G. W. 2010. Davinci: A cloud computing framework for service robots. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 3084–3089. (Cited on pages 30 y 121.)

AYDEMIR, A., PRONOBIS, A., GÖBELBECKER, M., AND JENSFELT, P. 2013. Active visual object search in unknown environments using uncertain semantics. *IEEE Transactions on Robotics 29,* 4, 986–1002. (Cited on page 56.)

BAKAMBU, J. N. AND POLOTSKI, V. 2007. Autonomous system for navigation and surveying in underground mines. *Field Robotics, Journal of 24,* 10 (July), 829–847. (Cited on page 87.)

BARRIOS, S., AYUSO, N., TARDIOLI, D., RIAZUELO, L., MOSTEO, A. R., LERA, F., AND VILLARROEL, J. L. 2017. Low-bandwidth telerobotics in fading scenarios. In *ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Vol. 2. (Cited on page 11.)

BAUDAT, G. AND ANOUAR, F. 2000. Generalized discriminant analysis using a kernel approach. *Neural Comput. 12,* 10 (Oct.), 2385–2404. (Cited on page 94.)

BAY, H., ESS, A., TUYTELAARS, T., AND VAN GOOL, L. 2008. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding 110,* 3, 346–359. (Cited on pages 15 y 60.)

BAY, H., TUYTELAARS, T., AND VAN GOOL, L. 2006. Surf: Speeded up robust features. In *Computer Vision-ECCV 2006*. Vol. 3951. 404–417. (Cited on page 119.)

BEETZ, M., MÖSENLECHNER, L., AND TENORTH, M. 2010. CRAM – A Cognitive Robot Abstract Machine for Everyday Manipulation in Human Environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. (Cited on page 56.)

BERGTHOLDT, M., KAPPES, J., SCHMIDT, S., AND SCHNÖRR, C. 2010. A study of parts-based object class detection using complete graphs. *Int. J. Comput. Vision 87,* 1-2 (Mar.), 93–117. (Cited on page 75.)

BIRK, A., SCHWERTFEGER, S., AND PATHAK, K. 2009. A networking framework for teleoperation in safety, security, and rescue robotics. *Wireless Commun. 16,* 1 (Feb.), 6–13. (Cited on page 88.)

BISTRY, H. AND ZHANG, J. 2010. A cloud computing approach to complex robot vision tasks using smart camera systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3195 –3200. (Cited on page 30.)

BLANCHARD, E., HARZALLAH, M., BRIAND, H., AND KUNTZ, P. 2005. A typology of ontology-based semantic measures. In *EMOI-INTEROP workshop at the 17th Conf. on Advanced Information Systems Engineering*. Porto, Portugal. (Cited on page 57.)

BOOIJ, O., TERWIJN, B., ZIVKOVIC, Z., AND KROSE, B. 2007. Navigation using an appearance based topological map. In *Robotics and Automation, 2007 IEEE International Conference on*. 3927–3932. (Cited on page 87.)

BURGARD, W., MOORST, M., FOX, D., SIMMONS, R., AND THRUN, S. 2000. Collaborative Multi-Robot Exploration. In *2000 International Conference on Robotics and Automation*. San Francisco, USA, 476–481. (Cited on pages 87 y 89.)

CADENA, C., GÁLVEZ, D., RAMOS, F., TARDÓS, J., AND NEIRA, J. 2010. Robust place recognition with stereo cameras. Taipei, Taiwan. (Cited on page 14.)

CASTLE, R., KLEIN, G., AND MURRAY, D. 2010. Combining monoSLAM with object recognition for scene augmentation using a wearable camera. *Image and Vision Computing 28,* 11, 1548–1556. (Cited on pages 14, 55 y 75.)

CASTLE, R., KLEIN, G., AND MURRAY, D. 2011. Wide-area augmented reality using camera tracking and mapping in multiple regions. *Computer Vision and Image Understanding*. (Cited on page 30.)

CASTLE, R. O., GAWLEY, D. J., KLEIN, G., AND MURRAY, D. W. 2007. Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 4102–4107. (Cited on page 14.)

CASTLE, R. O., KLEIN, G., AND MURRAY, D. W. 2008. Video-rate localization in multiple maps for wearable augmented reality. In *12th IEEE International Symposium on Wearable Computers*. 15–22. (Cited on page 30.)

CHURCHILL, W. AND NEWMAN, P. 2012. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *IEEE International Conference on Robotics and Automation (ICRA2012)*. (Cited on page 29.)

CIVERA, J., DAVISON, A., AND MONTIEL, J. 2008a. Interacting multiple model monocular SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*. 3704–3709. (Cited on page 32.)

CIVERA, J., DAVISON, A. J., AND MONTIEL, J. M. M. 2008b. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics 24,* 5 (October), 932–945. (Cited on page 19.)

CIVERA, J., GÁLVEZ-LÓPEZ, D., RIAZUELO, L., TARDÓS, J. D., AND MONTIEL, J. M. M. 2011. Towards semantic slam using a monocular camera. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 1277 –1284. (Cited on pages 10, 29, 51, 55, 60, 75 y 119.)

CIVERA, J., GRASA, O. G., DAVISON, A. J., AND MONTIEL, J. M. M. 2010. 1-point ransac for ekf filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics 27,* 5 (October), 609–631. (Cited on pages 14 y 18.)

CLEMENTE, L. A., DAVISON, A. J., REID, I. D., NEIRA, J., AND TARDOS, J. D. 2007. Mapping large loops with a single hand-held camera. In *Robotics: Science and Systems*. (Cited on page 35.)

CUMMINS, M. AND NEWMAN, P. 2008. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research 27,* 6, 647. (Cited on pages 14 y 87.)

DIXON, C. AND FREW, E. W. 2009. Maintaining Optimal Communiaction Chains in Robotic Sensors Networks Using Mobility Control. *Mobile Networks and Applications 14,* 3, 281–291. (Cited on page 88.)

DONDRUP, C., BELLOTTO, N., JOVAN, F., AND HANHEIDE, M. 2015. Real-time multisensor people tracking for human-robot spatial interaction. In *Workshop on Machine Learning for Social Robotics at International Conference on Robotics and Automation (ICRA)*. ICRA/IEEE. (Cited on page 75.)

DURRANT-WHYTE, H. AND BAILEY, T. 2006. Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *Robotics and Automation Magazine 13,* 2, 99–110. (Cited on page 30.)

EKVALL, S., JENSFELT, P., AND KRAGIC, D. 2006. Integrating active mobile robot object recognition and slam in natural environments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5792–5797. (Cited on page 14.)

EKVALL, S. AND KRAGIC, D. 2005. Receptive field cooccurrence histograms for object detection. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 84–89. (Cited on page 55.)

ESS, A., LEIBE, B., SCHINDLER, K., AND VAN GOOL, L. 2009. Robust multiperson tracking from a mobile platform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 31,* 10, 1831–1846. (Cited on page 75.)

FERRARI, V., JURIE, F., AND SCHMID, C. 2010. From images to shape models for object detection. *International journal of computer vision 87,* 3, 284–303. (Cited on page 25.)

FOX, D. 2003. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research 22,* 12, 985–1003. (Cited on pages 89 y 91.)

FURUKAWA, Y., CURLESS, B., SEITZ, S., AND SZELISKI, R. 2010. Towards internet-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*. 1434–1441. (Cited on page 50.)

FURUKAWA, Y. AND PONCE, J. 2010. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32,* 8, 1362–1376. (Cited on page 16.)

GALINDO, C., FERNÁNDEZ-MADRIGAL, J.-A., GONZÁLEZ, J., AND SAFFIOTTI, A. 2008. Robot task planning using semantic maps. *Robotics and Autonomous Systems 56,* 11, 955–966. (Cited on page 55.)

GALLUP, D., FRAHM, J., AND POLLEFEYS, M. 2010. Piecewise planar and non-planar stereo for urban scene reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1418–1425. (Cited on page 50.)

GÁLVEZ-LÓPEZ, D., SALAS, M., TARDÓS, J. D., AND MONTIEL, J. 2016. Real-time monocular object slam. *Robot. Auton. Syst. 75,* PB (Jan.), 435–449. (Cited on page 75.)

GÁLVEZ-LÓPEZ, D. AND TARDÓS, J. D. 2012. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics 28,* 5, 1188–1197. (Cited on pages 68 y 79.)

GAUGLITZ, S., SWEENEY, C., VENTURA, J., TURK, M., AND HOLLERER, T. 2012. Live tracking and mapping from both general and rotation-only camera motion. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*. IEEE, 13–22. (Cited on page 32.)

GOERKE, N. AND BRAUN, S. 2009. Building semantic annotated maps by mobile robots tracking. *Towards Autonomous Robotic Systems*, 149–156. (Cited on page 75.)

GOLDBERG, K. AND KEHOE, B. 2013. Cloud robotics and automation: A survey of related work. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5*. (Cited on pages 27 y 121.)

GOOGLE. 2011. 3d warehouse. *Available at* `http://sketchup.google.com/3dwarehouse/`. (Cited on page 117.)

GOULD, S., FULTON, R., AND KOLLER, D. 2010. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 1–8. (Cited on page 25.)

GRANSTRÖM, K., SCHÖN, T. B., NIETO, J. I., AND RAMOS, F. T. 2011. Learning to close loops from range data. *The International Journal of Robotics Research 30,* 14, 1728–1754. (Cited on page 87.)

GRISETTI, G., KUMMERLE, R., STACHNISS, C., AND BURGARD, W. 2010. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE 2,* 4 (winter), 31–43. (Cited on pages 87 y 89.)

GRISETTI, G., STACHNISS, C., AND BURGARD, W. 2005. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. 2432–2437. (Cited on page 91.)

GRISETTI, G., STACHNISS, C., AND BURGARD, W. 2007. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics 23,* 1 (February), 34–46. (Cited on page 60.)

GRISETTI, G., TIPALDI, G. D., STACHNISS, C., BURGARD, W., AND NARDI, D. 2007. Fast and accurate {SLAM} with rao–blackwellized particle filters. *Robotics and Autonomous Systems 55,* 1, 30 – 38. Simultaneous Localisation and Map Building. (Cited on page 91.)

GUIZZO, E. 2011. Robots with their heads in the clouds. *IEEE Spectrum 48,* 3, 16–18. (Cited on page 30.)

GÜNTHER, M., WIEMANN, T., ALBRECHT, S., AND HERTZBERG, J. 2013. Building semantic object maps from sparse and noisy 3d data. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2228–2233. (Cited on pages 55 y 75.)

GUPTA, R. AND KOCHENDERFER, M. J. 2004. Common sense data acquisition for indoor mobile robots. In *Nineteenth National Conf. on Artificial Intelligence (AAAI-04)*. 605–610. (Cited on page 62.)

HABIB, M., BAUDOIN, Y., AND NAGATA, F. 2011. Robotics for rescue and risky intervention. In *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*. 3305–3310. (Cited on page 87.)

HARNAD, S. 1990. The symbol grounding problem. *Physica D 42*, 335–346. (Cited on page 54.)

HARTLEY, R. I. AND ZISSERMAN, A. 2004. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518. (Cited on pages 16 y 18.)

HERMANS, A., FLOROS, G., AND LEIBE, B. 2014. Dense 3d semantic mapping of indoor scenes from RGB-D images. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. 2631–2638. (Cited on page 75.)

HINTERSTOISSER, S., LEPETIT, V., ILIC, S., FUA, P., AND NAVAB, N. 2010. Dominant orientation templates for real-time detection of texture-less objects. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2257 –2264. (Cited on page 14.)

HOIEM, D., EFROS, A., AND HEBERT, M. 2007. Recovering surface layout from an image. *International Journal of Computer Vision 75,* 1, 151–172. (Cited on page 29.)

HORNUNG, A., WURM, K. M., BENNEWITZ, M., STACHNISS, C., AND BURGARD, W. 2013. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Auton. Robots 34,* 3 (Apr.), 189–206. (Cited on pages 29, 50, 61, 73, 75 y 79.)

HOWARD, A. 2006. Multi-robot Simultaneous Localization and Mapping using Particle Filters. *The International Journal of Robotics Research 25*, 1243–1256. (Cited on pages 87 y 89.)

HUBER, P. J. 1981. *Robust Statistics*. Wiley-Interscience. (Cited on page 77.)

HUNZIKER, D., GAJAMOHAN, M., WAIBEL, M., AND D'ANDREA, R. 2013. Rapyuta: The RoboEarth cloud engine. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), Karlsruhe, Germany*. 438–444. (Cited on pages 60 y 121.)

JAFARI, O. H., MITZEL, D., AND LEIBE, B. 2014. Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*. 5636–5643. (Cited on pages 76, 77, 79, 80 y 81.)

JOHO, D., SENK, M., AND BURGARD, W. 2011. Learning search heuristics for finding objects in structured environments. *Robotics and Autonomous Systems 59,* 5 (May), 319–328. (Cited on page 55.)

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT). 2011. Kit object models web database. *Available at* `http://i61p109.ira.uka.de/ObjectModelsWebUI/`. (Cited on page 117.)

KATO, H. AND BILLINGHURST, M. 1999. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Augmented Reality, 1999. (IWAR '99) Proceedings. 2nd IEEE and ACM International Workshop on*. 85–94. (Cited on page 118.)

KEHOE, B., PATIL, S., ABBEEL, P., AND GOLDBERG, K. 2015. A survey of research on cloud robotics and automation. *IEEE T-ASE Special Issue on Cloud Robotics and Automation 12,* 2. (Cited on page 53.)

KEHOE, B., WARRIER, D., PATIL, S., AND GOLDBERG, K. 2015. Cloud-based grasp planning for toleranced parts using parallelized monte carlo sampling. *IEEE T-ASE Special Issue on Cloud Robotics and Automation 12,* 2. (Cited on page 53.)

KIM, B., KAESS, M., FLETCHER, L., LEONARD, J., BACHRACH, A., ROY, N., AND TELLER, S. 2010. Multiple relative pose graphs for robust cooperative mapping. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 3185–3192. (Cited on page 29.)

KLEIN, G. AND MURRAY, D. 2007. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM Int. Symposium on*. 225–234. (Cited on pages 27, 30, 32, 39, 76 y 78.)

KLEIN, G. AND MURRAY, D. 2008. Improving the Agility of Keyframe-Based SLAM. In *Proceedings of the 10th European Conference on Computer Vision: Part II*. Springer, 802–815. (Cited on pages 31 y 34.)

KLEIN, G. AND MURRAY, D. 2009. Parallel tracking and mapping on a camera phone. In *8th IEEE International Symposium on Mixed and Augmented Reality*. 83–86. (Cited on page 41.)

KOCH, A., MARCO, D. D., WINLKER, J., AND HÄUSSERMANN, K. 2011. Recording and storing in the roboearth database. (Cited on page 118.)

KOENIG, N. AND HOWARD, A. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2149–2154. (Cited on page 66.)

KOLLAR, T. AND ROY, N. 2009. Utilizing object-object and object-scene context when planning to find things. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE, 2168–2173. (Cited on page 55.)

KRUSE, T., PANDEY, A. K., ALAMI, R., AND KIRSCH, A. 2013. Human-aware robot navigation: A survey. *Robot. Auton. Syst. 61,* 12 (Dec.), 1726–1743. (Cited on page 75.)

KUIPERS, B., MODAYIL, J., BEESON, P., MACMAHON, M., AND SAVELLI, F. 2004. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Vol. 5. 4845–4851 Vol.5. (Cited on page 87.)

KUNZE, L., BEETZ, M., SAITO, M., AZUMA, H., OKADA, K., AND INABA, M. 2012. Searching objects in large-scale indoor environments: A decision-thereotic approach. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*. St. Paul, MN, USA. (Cited on page 55.)

KUNZE, L., DORESWAMY, K. K., AND HAWES, N. 2014. Indirect object search based on qualitative spatial relations. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*. Hong Kong, China. Accepted for publication. (Cited on page 55.)

KUNZE, L., ROEHM, T., AND BEETZ, M. 2011. Towards semantic robot description languages. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*. Shanghai, China, 5589–5595. (Cited on page 53.)

KUNZE, L., TENORTH, M., AND BEETZ, M. 2010. Putting People's Common Sense into Knowledge Bases of Household Robots. In *33rd Annual German Conf. on Artificial Intelligence (KI 2010)*. Springer, Karlsruhe, Germany, 151–159. (Cited on page 62.)

LAI, K., BO, L., REN, X., AND FOX, D. 2011. A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE International Conference on Robotics & Automation (ICRA)*. (Cited on page 29.)

LATOMBE, J.-C. 1991. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA. (Cited on page 59.)

LAZARO, M. T., URCOLA, P., CASTELLANOS, J., AND MONTANO, L. 2012. Position Tracking and Path Planning in Uncertain Maps for Robot Formations. In *2nd IFAC Workshop on Multivehicle Systems*. Espoo, Finland. (Cited on pages 87 y 89.)

LEIBE, B., SEEMANN, E., AND SCHIELE, B. 2005. Pedestrian detection in crowded scenes. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. IEEE Computer Society, Washington, DC, USA, 878–885. (Cited on page 75.)

LITOMISKY, K. AND BHANU, B. 2012. Removing moving objects from point cloud scenes. X. Jiang, O. R. P. Bellon, D. B. Goldgof, and T. Oishi, Eds. Lecture Notes in Computer Science, vol. 7854. Springer, 50–58. (Cited on page 75.)

LUNENBURG, J., VAN DEN DRIES, S., ELFRING, J., JANSSEN, R., SANDEE, J., AND VAN DE MOLENGRAFT, M. 2012. Tech united eindhoven team description 2012. In *RoboCup Team Description Papers 2012*. (Cited on page 66.)

MARCO, D. D., KOCH, A., ZWEIGLE, O., HÄUSSERMANN, K., SCHIESSLE, B., LEVI, P., GÁLVEZ-LÓPEZ, D., RIAZUELO, L., CIVERA, J., MONTIEL, J. M. M., TENORTH, M., PERZYLO, A. C., WAIBEL, M., AND VAN DE MOLENGRAFT, R. 2012. Creating and using roboearth object models. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*. 3549–3550. (Cited on page 11.)

MARINAKIS, D. AND DUDEK, G. 2010. Pure topological mapping in mobile robotics. *Robotics, IEEE Transactions on 26,* 6 (Dec), 1051–1064. (Cited on page 87.)

MARTINEZ, M., COLLET, A., AND SRINIVASA, S. 2010. Moped: A scalable and low latency object recognition and pose estimation system. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. 2043 –2049. (Cited on page 14.)

MASON, J. AND MARTHI, B. 2012. An object-based semantic world model for long-term change detection and semantic querying. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 3851–3858. (Cited on page 55.)

MEGER, D., FORSSÉN, P.-E., LAI, K., HELMER, S., MCCANN, S., SOUTHEY, T., BAUMANN, M., LITTLE, J. J., AND LOWE, D. G. 2008. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems 56,* 6 (June), 503–511. (Cited on pages 14 y 55.)

MICROSOFT. 2011. Kinect camera. *Available at* `http://www.xbox.com/kinect`. (Cited on page 117.)

MIKOLAJCZYK, K., LEIBE, B., AND SCHIELE, B. 2006. Multiple object class detection with a generative model. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*. Vol. 1. 26–36. (Cited on page 75.)

MILFORD, M. 2013. Vision-based place recognition: how low can you go? *The International Journal of Robotics Research 32,* 7, 766–789. (Cited on page 34.)

MINGUEZ, J. 2005. The obstacle-restriction method (orm) for robot obstacle avoidance in difficult environments. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*. (Cited on pages 60 y 91.)

MINGUEZ, J. AND MONTANO, L. 2004. Nearness diagram (nd) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation 20,* 1, 45–59. (Cited on pages 88 y 91.)

MINGUEZ, J., MONTANO, L., SIMÉON, T., AND ALAMI, R. 2001. Global nearness diagram navigation (gnd). In *IEEE Int. Conf. on Robotics and Automation (ICRA'01)*. 33–39. (Cited on page 91.)

MORENO-NOGUER, F., LEPETIT, V., AND FUA, P. 2007. Accurate non-iterative o(n) solution to the pnp problem. *Computer Vision, IEEE International Conference on 0*, 1–8. (Cited on pages 18 y 35.)

MÖSENLECHNER, L. AND BEETZ, M. 2011. Parameterizing Actions to have the Appropriate Effects. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. San Francisco, CA, USA, 4141–4147. (Cited on page 63.)

MOURIKIS, A. AND ROUMELIOTIS, S. 2006. Predicting the performance of cooperative simultaneous localization and mapping (c-slam). *The International Journal of Robotics Research 25,* 12, 1273–1286. (Cited on page 29.)

MOZOS, O. M., ROTTMANN, A., TRIEBEL, R., JENSFELT, P., AND BURGARD, W. 2007. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems 55*, 391–402. (Cited on page 87.)

MUJA, M. AND LOWE, D. G. 2009. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 331–340. (Cited on page 17.)

MUNARO, M., BASSO, F., AND MENEGATTI, E. 2016. Openptrack: Open source multi-camera calibration and people tracking for rgb-d camera networks. *Robot. Auton. Syst. 75,* PB (Jan.), 525–538. (Cited on page 80.)

MUR-ARTAL, R., MONTIEL, J. M. M., AND TARDOS, J. D. 2015. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics 31,* 5, 1147–1163. (Cited on pages 73, 76 y 78.)

MUR-ARTAL, R. AND TARDOS, J. D. 2016. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475. To appear in IEEE Trans. on Robotics.*. (Cited on pages 73, 75, 76, 78 y 80.)

MURPHY, K., TORRALBA, A., AND FREEMAN, W. 2003. Using the forest to see the trees: a graphical model relating features, objects and scenes. *Advances in Neural Information Processing Systems 16*. (Cited on page 25.)

MURPHY, R., KRAVITZ, J., STOVER, S., AND SHOURESHI, R. 2009. Mobile robots in mine rescue and recovery. *Robotics Automation Magazine, IEEE 16,* 2 (June), 91–103. (Cited on page 87.)

NEWCOMBE, R. AND DAVISON, A. 2010. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 1498–1505. (Cited on page 25.)

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHLI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality*. ISMAR '11. IEEE Computer Society, Washington, DC, USA, 127–136. (Cited on page 75.)

NEWMAN, P., COLE, D., AND HO, K. 2006. Outdoor slam using visual appearance and laser ranging. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. 1180–1187. (Cited on page 87.)

NGUYEN, H. G., FARRINGTON, N., AND PEZESHKIAN, N. 2004. Maintaining Communication Link for Tactical Ground Robots. In *AUVSI Unmanned System North America*. Anaheim, CA, USA. (Cited on page 88.)

NÜCHTER, A. AND HERTZBERG, J. 2008. Towards semantic maps for mobile robots. *Robot. Auton. Syst. 56,* 11 (Nov.), 915–926. (Cited on page 75.)

OLIVA, A., TORRALBA, A., CASTELHANO, M. S., AND HENDERSON, J. M. 2003. Top-down control of visual attention in object detection. In *Int. Conf. on Image Processing (ICIP)*. Vol. 1. I–253. (Cited on page 55.)

PANGERCIC, D., TENORTH, M., PITZER, B., AND BEETZ, M. 2012. Semantic object maps for robotic housework - representation, acquisition and use. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vilamoura, Portugal. (Cited on page 55.)

PEASGOOD, M., MCPHEE, J., AND CLARK, C. 2006. Complete and scalable multi-robot planning in tunnel environments. In *1st IFAC Workshop on Multivehicle Systems*. (Cited on page 87.)

QUIGLEY, M., CONLEY, K., GERKEY, B. P., FAUST, J., FOOTE, T., LEIBS, J., WHEELER, R., AND NG, A. Y. 2009. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*. (Cited on pages 39, 80, 98 y 117.)

RANGANATHAN, A. AND DELLAERT, F. 2007. Semantic modeling of places using objects. In *Robotics: Science and Systems*. (Cited on page 15.)

RANGANATHAN, A. AND DELLAERT, F. 2011. Online probabilistic topological mapping. *Int. J. Rob. Res. 30,* 6 (May), 755–771. (Cited on page 87.)

REMY, S. AND BLAKE, M. 2011. Distributed service-oriented robotics. *IEEE Internet Computing 15,* 2, 70–74. (Cited on page 30.)

RIAZUELO, L., CIVERA, J., AND MONTIEL, J. M. M. 2013. C^{2}TAM: A First Approach to a Cloud framework for Cooperative Tracking And Mapping. In *IROS 2013 Workshop on Cloud Robotics: Online Knowledge Bases, Web Services, and Cloud Computing for Robots*. Tokyo, Japan. (Cited on page 10.)

RIAZUELO, L., CIVERA, J., AND MONTIEL, J. M. M. 2014a. *Actas de la III Jornada de Jóvenes Investigadores del I3A*. Vol. 2. (Cited on page 10.)

RIAZUELO, L., CIVERA, J., AND MONTIEL, J. M. M. 2014b. C^{2}TAM: A Cloud framework for Cooperative Tracking And Mapping. *Robotics and Autonomous Systems 62,* 4 (April), 401–413. (Cited on pages 10, 60, 73, 75, 76, 78, 80 y 121.)

RIAZUELO, L., MONTANO, L., AND MONTIEL, J. M. 2017. Semantic visual slam in populated environments. In *The European Conference on Mobile Robotics (ECMR)*. Paris, France. (Cited on page 10.)

RIAZUELO, L., TENORTH, M., MARCO, D. D., SALAS, M., GÁLVEZ-LÓPEZ, D., MÖSENLECHNER, L., KUNZE, L., BEETZ, M., TARDÓS, J. D., MONTANO, L., AND MONTIEL, J. M. M. 2015. Roboearth semantic mapping: A cloud enabled knowledge-based approach. *IEEE Transactions on Automation Science and Engineering PP,* 99, 1–12. (Cited on page 10.)

RIAZUELO, L., TENORTH, M., MARCO, D. D., SALAS, M., MÖSENLECHNER, L., KUNZE, L., BEETZ, M., TARDOS, J. D., MONTANO, L., AND MONTIEL, J. M. M. 2013. Roboearth web-enabled and knowledge-based active perception. In *IROS 2013 Workshop on AI-based Robotics*. Tokyo, Japan. (Cited on page 10.)

RIZZO, C., SICIGNANO, D., RIAZUELO, L., TARDIOLI, D., LERA, F., VILLARROEL, J. L., AND MONTANO, L. 2015. Guaranteeing communication for robotic intervention in long tunnel scenarios. In *ROBOT 2015, Second Iberian Robotics Conference. Advances in Intelligent Systems and Computing.*, L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, Eds. Vol. 1. (Cited on page 11.)

RIZZO, C., TARDIOLI, D., SICIGNANO, D., RIAZUELO, L., VILLARROEL, J. L., AND MONTANO, L. 2013. Signal-based deployment planning for robot teams in tunnel-like fading environments. *The International Journal of Robotics Research 32,* 12, 1381–1397. (Cited on pages 11, 88, 89 y 107.)

ROBOEARTH CONSORTIUM. 2011. Roboearth. *Available at* `http://www.roboearth.org`. (Cited on pages 117 y 120.)

ROGERS, J., TREVOR, A., NIETO-GRANDA, C., AND CHRISTENSEN, H. 2011. Simultaneous localization and mapping with learned object recognition and semantic data association. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1264–1270. (Cited on page 30.)

ROMEO, A. AND MONTANO, L. 2006. Environment understanding: Robust feature extraction from range sensor data. In *IROS* (2009-05-11). IEEE, 3337–3343. (Cited on pages 89, 93 y 94.)

ROS ROBOEARTH. 2014. Ros packages for interfacing with roboearth. *Available at* `http://wiki.ros.org/roboearth_stack`. (Cited on page 64.)

ROSTEN, E. AND DRUMMOND, T. 2006. Machine learning for high-speed corner detection. *European Conference on Computer Vision (ECCV)*, 430–443. (Cited on pages 33 y 77.)

RUSSELL, B., TORRALBA, A., MURPHY, K., AND FREEMAN, W. 2008. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision 77,* 1, 157–173. (Cited on page 29.)

SABATTINI, L., CHOPRA, N., AND SECCHI, C. 2013. Decentralized connectivity maintenance for cooperative control of mobile robotic systems. *The International Journal of Robotics Research 32,* 12, 1411–1423. (Cited on page 88.)

SALAS, M. AND MONTIEL, J. 2011. Keyframe based semantic mapping. Tech. rep. December. (Cited on pages 29 y 51.)

SALAS-MORENO, R., NEWCOMBE, F. R. A., STRASDAT, H., KELLY, P. H., AND DAVISON, A. J. 2013. Slam++: Simultaneous localisation and mapping at the level of objects. In *IEEE Proc. Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 55 y 75.)

SALMERÓN-GARCÍA, J., DÍAZ-DEL RÍO, F., P., I.-B., AND D., C. 2015. A trade-off analysis of a cloud-based robot navigation assistant using stereo image processing. *IEEE T-ASE Special Issue on Cloud Robotics and Automation 12,* 2. (Cited on page 53.)

SCHIESSLE, B., HÄUSSERMANN, K., AND ZWEIGLE, O. 2011. Deliverable 6.1: Complete specification of the roboearth platform. technical report. (Cited on page 118.)

SCHUSTER, M., JAIN, D., TENORTH, M., AND BEETZ, M. 2012. Learning organizational principles in human environments. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*. St. Paul, MN, USA. (Cited on page 55.)

SHERMER, T. 1992. Recent results in art galleries. *Proc. IEEE 80,* 9, 1384 – 1399. (Cited on page 69.)

SHUBINA, K. AND TSOTSOS, J. K. 2010. Visual search for an object in a 3d environment using a mobile robot. *Computer Vision and Image Understanding 114,* 5, 535–547. (Cited on page 55.)

SNAVELY, N., SEITZ, S., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3D. In *ACM SIGGRAPH 2006 Papers*. ACM, 835–846. (Cited on page 60.)

SNAVELY, N., SEITZ, S., AND SZELISKI, R. 2008. Modeling the world from internet photo collections. *International Journal of Computer Vision 80,* 2, 189–210. (Cited on pages 14 y 16.)

STENMARK, M., MALEC, J., NILSSON, K., AND ROBERTSSON, A. 2015. On distributed knowledge bases for robotized small-batch assembly. *IEEE T-ASE Special Issue on Cloud Robotics and Automation 12,* 2. (Cited on page 53.)

STRASDAT, H., MONTIEL, J., AND DAVISON, A. 2010a. Real-time Monocular SLAM: Why Filter? In *IEEE International Conference on Robotics and Automation (ICRA)*. (Cited on pages 29 y 30.)

STRASDAT, H., MONTIEL, J., AND DAVISON, A. 2010b. Scale drift-aware large scale monocular SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*. (Cited on page 25.)

TARDIOLI, D., MOSTEO, A., RIAZUELO, L., VILLARROEL, J., AND MONTANO, L. 2010. Enforcing Network Connectivity in Robot Team Missions. *The International Journal of Robotics Research 29,* 4 (April), 460–480. (Cited on pages 11, 88 y 89.)

TARDIOLI, D., RIAZUELO, L., SECO, T., ESPELOSÍN, J., LALANA, J., VILLARROEL, J. L., AND MONTANO, L. 2017. A robotized dumper for debris removal in tunnels under construction. In *ROBOT 2017: Third Iberian Robotics Conference. Advances in Intelligent Systems and Computing*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds. Vol. 1. (Cited on page 11.)

TARDIOLI, D., SICIGNANO, D., RIAZUELO, L., ROMEO, A., VILLARROEL, J. L., AND MONTANO, L. 2016. Robot teams for intervention in confined and structured environments. *Journal of Field Robotics 33,* 6, 765–801. (Cited on pages 10 y 85.)

TARDIOLI, D., SICIGNANO, D., AND VILLARROEL, J. 2014. A wireless multi-hop protocol for real-time applications. *Computer Communications Available online 30 August 2014, ISSN 0140-3664, http://dx.doi.org/10.1016/j.comcom.2014.08.012*. (Cited on pages 89 y 94.)

TARDIOLI, D. AND VILLARROEL, J. L. 2007. Real time communications over 802.11: Rt-wmp. In *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*. 1–11. (Cited on page 95.)

TARDIOLI, D. AND VILLARROEL, J. L. 2014. Odomety-less localization in tunnel-like environments (to be published). In *IEEE International Conference on Autonomous Robot Systems and Competitions (IEEE ICARSC)*. IEEE. (Cited on page 92.)

TENORTH, M. AND BEETZ, M. 2009. Knowrob - knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4261–4266. (Cited on page 118.)

TENORTH, M. AND BEETZ, M. 2013. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. Part 1: The KnowRob System. *Int. Journal of Robotics Research 32,* 5, 566 – 590. (Cited on page 64.)

TENORTH, M., PERZYLO, A. C., LAFRENZ, R., AND BEETZ, M. 2013. Representation and Exchange of Knowledge about Actions, Objects, and Environments in the RoboEarth Framework. *IEEE Transactions on Automation Science and Engineering (T-ASE) 10,* 3, 643–651. (Cited on pages 53, 56 y 59.)

THRUN, S. AND MONTEMERLO, M. 2006. The graph slam algorithm with applications to large-scale mapping of urban structures. *Int. J. Rob. Res. 25,* 5-6 (May), 403–429. (Cited on page 87.)

TORRALBA, A. AND EFROS, A. 2011. Unbiased look at dataset bias. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 1521–1528. (Cited on page 29.)

TORRALBA, A., FERGUS, R., AND FREEMAN, W. T. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30,* 11, 1958–1970. (Cited on page 34.)

TRIGGS, B., MCLAUCHLAN, P., HARTLEY, R., AND FITZGIBBON, A. 2000. Bundle adjustment – A modern synthesis. In *Vision Algorithms: Theory and Practice*. LNCS. Springer Verlag, 298–375. (Cited on page 32.)

TULLY, S., KANTOR, G., AND CHOSET, H. 2012. A unified bayesian framework for global localization and slam in hybrid metric/topological maps. *The International Journal of Robotics Research*. (Cited on page 87.)

TULLY, S., KANTOR, G., CHOSET, H., AND WERNER, F. 2009. A multi-hypothesis topological slam approach for loop closing on edge-ordered graphs. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. 4943–4948. (Cited on page 87.)

VASUDEVAN, S. AND SIEGWART, R. 2008. Bayesian space conceptualization and place classification for semantic maps in mobile robotics. *Robotics and Autonomous Systems 56,* 6, 522–537. (Cited on page 55.)

VIOLA, P., JONES, M. J., AND SNOW, D. 2005. Detecting pedestrians using patterns of motion and appearance. *Int. J. Comput. Vision 63,* 2 (July), 153–161. (Cited on page 75.)

W3C. 2009. *OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax*. World Wide Web Consortium. http://www.w3.org/TR/2009/REC-owl2-syntax-20091027. (Cited on page 56.)

WAIBEL, M., BEETZ, M., D'ANDREA, R., JANSSEN, R., TENORTH, M., CIVERA, J., ELFRING, J., GALVEZ-LOPEZ, D., HAUSSERMANN, K., MONTIEL, J. M. M., PERZYLO, A., SCHIESSLE, B., ZWEIGLE, O., AND VAN DE MOLENGRAFT, R. 2010. Roboearth - a world wide web for robots. *IEEE Robotics and Automation Magazine*. Accepted for publication. (Cited on pages 23, 30, 53, 117 y 121.)

WALKER, V. 2009. Idaho national laboratory. In *INL Communications & Public Affairs*. (Cited on page 87.)

WANG, C., THORPE, C. E., THRUN, S., HEBERT, M., AND DURRANT-WHYTE, H. F. 2007. Simultaneous localization, mapping and moving object tracking. *I. J. Robotic Res. 26,* 9, 889–916. (Cited on page 75.)

WANG, Z., JENSFELT, P., AND FOLKESSON, J. 2015. Multi-scale conditional transition map: Modeling spatial-temporal dynamics of human movements with local and long-term correlations. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 6244–6251. (Cited on page 75.)

WANG, Z., JENSFELT, P., AND FOLKESSON, J. 2016. Building a human behavior map from local observations. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 64–70. (Cited on page 75.)

WENDEL, A., MAURER, M., GRABER, G., POCK, T., AND BISCHOF, H. 2012. Dense reconstruction on-the-fly. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 1450–1457. (Cited on page 30.)

WHITE, C., HIRANANDANI, D., OLSTAD, C. S., BUHAGIAR, K., GAMBIN, T., AND CLARK, C. M. 2010. The malta cistern mapping project: Underwater robot mapping and localization within ancient tunnel systems. *J. Field Robot. 27,* 4 (July), 399–411. (Cited on page 87.)

WILLIAMS, B., KLEIN, G., AND REID, I. 2007. Real-time SLAM relocalisation. In *IEEE 11th International Conference on Computer Vision*. 1:8. (Cited on pages 14 y 19.)

WILLOW GARAGE. 2011. Household objects database. *Available at* `http://www.ros.org/wiki/household_objects`. (Cited on page 117.)

WONG, L. L., KAELBLING, L. P., AND LOZANO-PÉREZ, T. 2013. Manipulation-based active search for occluded objects. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*. Karlsruhe, Germany, 2814–2819. (Cited on page 55.)

XIAO, S., WANG, Z., AND FOLKESSON, J. 2015. Unsupervised robot learning to predict person motion. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 691–696. (Cited on page 75.)

YAMAUCHI, B. 1997. A frontier-based approach for autonomous exploration. In *Proc. of the IEEE Int. Symposium on Computational Intelligence, Robotics and Automation*. 146–151. (Cited on page 60.)

YANG, J., JIN, Z., YU YANG, J., ZHANG, D., AND FRANGI, A. F. 2004. Essence of kernel fisher discriminant: {KPCA} plus {LDA}. *Pattern Recognition 37,* 10, 2097 – 2100. (Cited on page 94.)

YANG, Y. AND RAMANAN, D. 2011. Articulated pose estimation with flexible mixtures-of-parts. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*. 1385–1392. (Cited on page 75.)

ZENDER, H., MARTÍNEZ MOZOS, O., JENSFELT, P., KRUIJFF, G. J. M., AND BURGARD, W. 2008. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems 56,* 6 (June), 493–502. (Cited on pages 14, 55 y 87.)

ZHOU, K., ZILLICH, M., ZENDER, H., AND VINCZE, M. 2012. Web mining driven object locality knowledge acquisition for efficient robot behavior. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 3962–3969. (Cited on page 55.)

ZHUANG, F., ZUPAN, C., CHAO, Z., AND YANZHENG, Z. 2008. A cable-tunnel inspecting robot for dangerous environment. *5,* 3, 243–248. (Cited on page 87.)

ZLOT, R. AND BOSSE, M. 2014. Efficient Large-Scale Three-Dimensional Mobile Mapping for Underground Mines. *Field Robotics, Journal of 31*, 758–779. (Cited on page 87.)